MEC-SETEC
INSTITUTO FEDERAL MINAS GERAIS - *Campus* Formiga
Curso de Ciência da Computação

ANÁLISE COMPARATIVA DE TÉCNICAS DE CORRESPONDÊNCIA TEXTUAL DE PRODUTO: COLETA DE DADOS COM *WEB SCRAPING* E ANÁLISE BASEADA EM PLN E APRENDIZADO DE MÁQUINA

Rafael Augusto de Rezende Neto

Orientador: Everthon Valadão

Formiga - MG

2025

RAFAEL AUGUSTO DE REZENDE NETO

ANÁLISE COMPARATIVA DE TÉCNICAS DE CORRESPONDÊNCIA TEXTUAL DE PRODUTO: COLETA DE DADOS COM *WEB SCRAPING* E ANÁLISE BASEADA EM PLN E APRENDIZADO DE MÁQUINA

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Everthon Valadão

Formiga - MG

2025

Rezende Neto, Rafael Augusto de
R006.3a Análise cooperativa de técnicas de correspondência textual de produto: coleta de Dados com web scraping e análise baseada em PLN e aprendizagem de máquina. / Rafael Augusto de Rezende Neto – Formiga : IFMG, 2025.
94 p. :il. color.

Orientador: Prof. Me. Everthon Valadão
Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Resolução de entidades. 2. Processamento de linguagem natural (PLN). 3. Web scraping. 4. Correspondência Textual. 5. Coleta de dados I. Rezende Neto, Rafael Augusto de. II. Título.

CDD 006.3

Ficha catalográfica elaborada pela Bibliotecária Lívia Renata Santos- CRB/6-2561

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
Campus Formiga
Diretoria de Ensino
Docência Área Acadêmica de Computação
Rua São Luiz Gonzaga, s/n - Bairro São Luiz - CEP 35570-000 - Formiga - MG
- www.ifmg.edu.br

RAFAEL AUGUSTO DE REZENDE NETO

Análise Comparativa de Técnicas de Correspondência Textual de Produto: coleta de dados com *web scraping* e análise baseada em PLN e aprendizado de máquina

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

APROVADO em: 10 de dezembro de 2025.

BANCA EXAMINADORA

Prof.º Me. Everthon Valadão dos Santos (orientador, IFMG)

Prof.º Me. Diego Mello da Silva (coorientador, IFMG)

Prof.º Dr. Bruno Ferreira (IFMG)

Prof.º Me. Mário Luiz Rodrigues Oliveira (IFMG)

Prof.º Me. Walace de Almeida Rodrigues (IFMG)

Documento assinado eletronicamente por **Bruno Ferreira**, **Professor**, em 10/12/2025, às 16:20, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Everthon Valadao dos Santos**, **Professor**, em 10/12/2025, às 16:21, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Diego Mello da Silva**, **Professor**, em 10/12/2025, às 16:21, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Mario Luiz Rodrigues Oliveira**, **Professor**, em 10/12/2025, às 16:21, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Walace de Almeida Rodrigues**, **Professor**, em 10/12/2025, às 16:22, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **2538979** e o código CRC **F1128A4F**.

Criado por everthon.valadao, versão 4 por everthon.valadao em 27/11/2025 11:16:14.

Agradecimentos

Agradeço a Deus pela sabedoria, força e proteção concedidas ao longo de toda esta jornada. À minha esposa, Marielle, e à minha filha, Maria, pelo amor, apoio constante e compreensão diante dos momentos de ausência. Aos meus pais, Rafael e Edna, pelo incentivo, exemplo de dedicação e pelos valores que me trouxeram até aqui. Ao meu orientador, Everthon Valadão, e coorientador, Diego Melo, pela orientação técnica, confiança, paciência e contribuições fundamentais para a construção deste trabalho.

"Dedico este trabalho a minha família, que sempre esteve ao meu lado, me ajudando e motivando, mesmo nos momentos mais difíceis."

Resumo

O crescimento acelerado do comércio eletrônico brasileiro, especialmente no setor de supermercados, tem impulsionado a necessidade de ferramentas automatizadas para comparação de produtos entre múltiplas plataformas. Nesse cenário, a correspondência textual de produtos torna-se um desafio central, devido à ausência de identificadores únicos e à heterogeneidade das descrições — frequentemente ruidosas, semiestruturadas e inconsistentes. Assim, justificou-se a investigação de técnicas de Processamento de Linguagem Natural (PLN) e aprendizado de máquina, capazes de lidar com esse ambiente real de dados, visando aprimorar a Resolução de Entidades (*Entity Resolution*) no domínio do varejo alimentar. O objetivo deste trabalho foi analisar comparativamente o desempenho de seis técnicas de correspondência textual — três clássicas (*Levenshtein*, *Jaccard* e *Jaro-Winkler*), duas vetoriais (*Bag-of-Words* e TF-IDF com Similaridade de Cossenos) e uma semântica (SBERT) — aplicadas a um *dataset* real coletado via *web scraping* em quatro fontes distintas de *e-commerce*. O *pipeline* envolveu coleta, pré-processamento e aplicação das técnicas, seguido da avaliação quantitativa baseada em Precisão, Revocação e *F1-Score*, com validação por *ground truth* gerado. Os resultados indicaram que as técnicas clássicas e vetoriais obtiveram desempenho superior no *matching* exato, enquanto o modelo semântico SBERT apresentou limitações, sobretudo pela ausência de *fine-tuning* e pela ocorrência de *alucinações* semânticas, como demonstrado na análise de resultados. Dessa forma, a hipótese inicial — de que a técnica semântica seria a mais eficaz — foi parcialmente refutada, embora apontando caminhos promissores para adaptações futuras. Conclui-se que os objetivos foram plenamente alcançados e que os métodos léxicos e vetoriais ainda são mais adequados para problemas de *matching* preciso em dados de supermercado. As principais contribuições incluem a criação de um *dataset* real, um pipeline replicável de análise prática e a avaliação crítica das técnicas, fornecendo subsídios para soluções híbridas e estudos futuros com adaptação de domínio.

Palavras-chave: Resolução de Entidades, Processamento de Linguagem Natural (PLN), *Web Scraping*

Abstract

The rapid growth of Brazilian e-commerce — particularly within the supermarket sector — has intensified the demand for automated tools capable of comparing products across multiple platforms. In this context, textual product matching emerges as a central challenge due to the absence of unique identifiers and the high heterogeneity of product descriptions, which are frequently noisy, semi-structured, and inconsistent. Therefore, this study justifies the investigation of Natural Language Processing (NLP) and machine learning techniques capable of handling real-world data environments, aiming to improve Entity Resolution in the grocery retail domain. The objective of this work was to perform a comparative analysis of the performance of six textual matching techniques — three classical (Levenshtein, Jaccard, and Jaro-Winkler), two vector-based (Bag-of-Words and TF-IDF with Cosine Similarity), and one semantic approach (SBERT) — applied to a real dataset collected via web scraping from four distinct e-commerce sources. The methodology encompassed data collection, preprocessing, implementation of the techniques, and quantitative evaluation based on Precision, Recall, and F1-Score, validated through a manually generated ground truth. The results showed that classical and vector-based techniques achieved superior performance for exact matching tasks, while the semantic SBERT model presented limitations, mainly due to the absence of fine-tuning and the occurrence of "semantic hallucinations," as evidenced in the results. Consequently, the initial hypothesis — that the semantic technique would be the most effective — was partially refuted, although showing promising directions for future adaptations. It is concluded that all objectives were successfully achieved, and that lexical and vector-based methods remain more suitable for precise matching problems in supermarket datasets. The main contributions include the creation of a real dataset, a replicable practical pipeline, and a critical evaluation of the methods — offering valuable insights for hybrid solutions and future research involving domain adaptation.

Keywords: Entity Resolution, Natural Language Processing (NLP), Web Scraping

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Aplication Programming Interface*
URL	*Uniform Resource Locator*
RAM	*Random Access Memory*
CPU	*Central Processing Unit*
TCC	Trabalho de Conclusão de Curso
IDE	*Integrated Development Environment*
PLN	Processamento de Linguagem Natural
ER	*Entity Resolution*
BoW	*Bag-of-Words*
TF-IDF	*Term Frequency-Inverse Document Frequency*
CAGR	*Compound Annual Growth Rate*
LLM	*Large Language Model*
BERT	*Bidirectional Encoder Representations from Transformers*
SBERT	*Sentence Bidirectional Encoder Representations from Transformers*
HTML	*Hypertext Markup Language*
CSS	*Cascading Style Sheets*
CSV	*Comma-Separated Values*
JSON	*JavaScript Object Notation*
DOM	*Document Object Model*
IP	*Internet Protocol*
LGPD	Lei Geral de Proteção de Dados Pessoais
TP	*True Positive*
FP	*False Positive*

FN	*False Negative*
TN	*True Negative*
GMC	*Google Merchant Center*
ASIN	*Amazon Standard Identification Number*
SQL	*Structured Query Language*
WIP	*Work in Progress*

Sumário

1 Introdução

O comércio eletrônico (*e-commerce*) tem reconfigurado radicalmente as dinâmicas de consumo e concorrência em escala global. No Brasil, este setor não apenas apresenta crescimento contínuo, mas o faz em um ritmo exponencial, criando um ecossistema digital de vastas proporções e complexidade. Dados recentes indicam que o faturamento do *e-commerce* brasileiro atingiu U$ 17,8 bilhões apenas no primeiro semestre de 2025. Projeções de mercado consolidam essa tendência, estimando que o volume do comércio eletrônico no país alcance US$ 586 bilhões até 2027, impulsionado por uma robusta taxa composta de crescimento anual (CAGR) de 19% para o período (LUCIO, 2025; PaymentsCMI, 2025). Este cenário é corroborado por um amadurecimento no comportamento do consumidor, que cada vez mais utiliza canais digitais para suas jornadas de compra. Um indicador notável é o recorde histórico na participação das buscas orgânicas, que alcançaram 29,5% do tráfego total no *e-commerce*, demonstrando uma maior intencionalidade e maturidade do comprador (IVO, 2025).

Contudo, este crescimento macroeconômico não é monolítico. Análises setoriais revelam que, enquanto alguns segmentos tradicionais passam por retrações (IVO, 2025), o setor de alimentos e supermercados emerge como um dos principais protagonistas da expansão digital (COSTA, 2025). De acordo com dados da Nielsen, a categoria de alimentos registrou um expressivo crescimento de 18,4% no faturamento bruto no primeiro semestre de 2024. Este avanço foi impulsionado pela crescente demanda por itens de giro rápido, indicando que as cestas de compras contendo Alimentos e Bebidas foram as principais impulsionadoras de crescimento do comércio eletrônico como um todo, representando 51% do total de pedidos (AKIRA, 2025).

Esta migração dos supermercados para o ambiente *online*, consolidando o *e-commerce* de *grocery* (COSTA, 2025), gera um desafio de dados que é central para esta pesquisa. Diferente de setores como eletrônicos, onde os produtos possuem SKUs (*Stock Keeping Units*) e atributos bem definidos, o domínio de supermercados é caracterizado por um volume massivo de produtos com baixa diferenciação textual, alta frequência de transações e descrições altamente variáveis. Para consumidores, a capacidade de comparar produtos idênticos — seja para encontrar o melhor preço ou avaliar a disponibilidade — e para empresas — seja para analisar a concorrência ou realizar estudos de mercado — torna-se uma necessidade (SOVIERSOVSKI, 2025; PHOENIX, 2024).

A consolidação de múltiplas plataformas (varejistas, *marketplaces*, aplicativos de entrega) leva o consumidor a consultar diversas fontes antes de uma decisão. É precisamente esta arquitetura de mercado — fragmentada em múltiplas fontes de dados — que instancia

o problema central deste trabalho: a dificuldade em determinar, de forma automática e escalável, que um "Leite Integral Longa Vida 1 Litro" em um *website* e um "Leite UHT 1L" em outro são, de fato, a mesma entidade do mundo real.

A dificuldade exposta é um problema clássico e central na ciência da computação e na ciência de dados, formalmente conhecido como Resolução de Entidades (ER — *Entity Resolution*). A literatura acadêmica utiliza, por vezes de forma intercambiável, termos como *record linkage* (ligação de registros), *fuzzy matching* (correspondências aproximadas) ou, especificamente neste domínio, *product matching* (correspondência de produtos) (RIVAS-SáNCHEZ et al., 2017; MARCHANT; RUBINSTEIN; STEORTS, 2023). A Resolução de Entidades é formalmente definida como o processo de identificar e agrupar registros em um ou mais bancos de dados que se referem à mesma entidade do mundo real (MüLLER; KUWERTZ, 2022). Como observado na premissa desta monografia, o desafio é particularmente agudo "na ausência de um identificador único" (MARCHANT; RUBINSTEIN; STEORTS, 2023).

No contexto específico do comércio eletrônico, o problema da Resolução de Entidades é exacerbado por uma série de fatores. As fontes de dados não são apenas múltiplas, mas fundamentalmente heterogêneas, ruidosas e, muitas vezes, semiestruturadas. Os registros de produtos (as "entidades") são caracterizados por descrições textuais (títulos, especificações, atributos) que variam drasticamente em formato, terminologia e nível de detalhe entre diferentes vendedores. Além disso, os dados são frequentemente esparsos, com atributos-chave como "marca" ou "unidades de medida" ausentes na maioria dos registros, tornando a correspondência baseada puramente em atributos estruturados impraticável (YULIANTON; SANTI, 2024).

Portanto, a tarefa de identificar automaticamente produtos idênticos recai quase inteiramente sobre a análise de suas descrições textuais. Isso posiciona o Processamento de Linguagem Natural (PLN) e as técnicas de similaridade de texto no centro da solução (MüLLER; KUWERTZ, 2022). O campo de pesquisa em ER, longe de estar resolvido, evolui rapidamente. Pesquisas recentes focam em superar as limitações de métodos tradicionais, explorando desde o aprendizado de máquina supervisionado até, mais recentemente, a eficácia de grandes modelos de linguagem (LLMs) e modelos de linguagem pré-treinados (PLMs), como o BERT, em tarefas de correspondência (PEETERS; STEINER; BIZER, 2025).

Este projeto propõe investigar, implementar e analisar diferentes técnicas e algoritmos focados na correspondência (ou correlação) de produtos baseada em suas descrições textuais. O escopo abrange desde a identificação de fontes de dados relevantes (*websites* de *e-commerce*) e a construção de mecanismos de coleta de dados (*web scraping*), passando pela necessária etapa de limpeza e organização das informações textuais, até a aplicação e avaliação comparativa de métodos de processamento de linguagem natural (PLN) e

similaridade de texto para identificar produtos que representam a mesma entidade no mundo real, embora descritos de formas distintas (SOVIERSOVSKI, 2025; PHOENIX, 2024). O objetivo final é determinar abordagens eficazes que possam servir de base para sistemas de comparação de preços ou análise de sortimento de produtos.

1.1 Justificativa

A capacidade de coletar, processar e comparar dados de produtos em larga escala não é apenas uma vantagem competitiva, mas uma necessidade operacional. A relevância deste projeto de pesquisa, focado na "Análise Comparativa de Técnicas de Correspondência Textual de Produto", justifica-se, portanto, em múltiplas dimensões que se interconectam: a prática (mercado), a acadêmica (científica) e a tecnológica.

A relevância prática do projeto é imediata, atendendo a demandas urgentes tanto do consumidor quanto das empresas.

Do ponto de vista do consumidor, a demanda por ferramentas precisas de comparação de preços é um pilar do *e-commerce* moderno. Conforme apontado por Fernandes (2023), *marketplaces* (47,8%) e portais de comparação de preços (47,1%) já figuravam como as principais referências de consumo. O advento de assistentes de compra baseados em inteligência artificial (E-Commerce Update, 2025) eleva essa expectativa a um novo patamar, exigindo uma precisão de correspondência de produtos que sistemas legados não conseguem fornecer. A falha em identificar corretamente que "Leite Integral Longa Vida 1 Litro" e "Leite UHT 1L" são o mesmo item resulta em comparações inúteis, frustrando o consumidor e erodindo a confiança na plataforma.

Para as empresas, a correspondência de produtos (*product matching*) é a infraestrutura de dados que sustenta decisões estratégicas de alto impacto. Conforme destacado pela indústria de inteligência de varejo (BOER, 2024), o *product matching* é o *input* essencial para equipes de precificação, permitindo o monitoramento em tempo real de preços, promoções e estoque concorrente. Sem uma correspondência precisa, a implementação de estratégias de precificação dinâmica (*dynamic pricing*) (Pricer24, 2023; TGNData, 2024) torna-se inviável ou arriscada. Uma correspondência falha pode levar uma empresa a reduzir seu preço para competir com um produto inferior (corroendo a margem) ou falhar em identificar um concorrente direto (perdendo vendas).

Além da precificação, a análise de sortimento (*assortment analysis*) — a compreensão de quais produtos os concorrentes oferecem e quais lacunas existem no próprio catálogo — depende dessa mesma tecnologia (NOVKOVIC, 2025). O problema central que este TCC ataca é a "heterogeneidade dos dados" que impede as empresas de alcançarem um "Registro Dourado" (*Golden Record*) — uma visão unificada, completa e confiável de seus produtos e do mercado (BEACH, 2021). A má qualidade dos dados, resultante de falhas

na resolução de entidades, gera um custo financeiro direto, estimado pelo Gartner em $12,9 milhões anuais por empresa (Equifax, 2023). Este trabalho busca, portanto, avaliar as ferramentas técnicas que mitigam esse prejuízo.

A relevância acadêmica do projeto reside em sua abordagem a um desafio clássico da Ciência de Dados: a Resolução de Entidades (ER), ou *product matching* (CHRISTOPHIDES; EFTHYMIOU; STEFANIDIS, 2015; ELMAGARMID; IPEIROTIS; VERYKIOS, 2007). Este é um problema de pesquisa ativo devido à sua complexidade computacional quadrática, $O(n^2)$ (PAPADAKIS et al., 2016), que exige soluções eficientes para a comparação de milhões de produtos em catálogos de *e-commerce* (PRIMPELI; BIZER, 2022). Esta pesquisa contribui ao focar na fase de *matching*, analisando o *trade-off* (relação custo-benefício) entre a eficiência de algoritmos léxicos (ex: Levenshtein), vetoriais (ex: TF-IDF) e modelos semânticos profundos (ex: SBERT) (REIMERS; GUREVYCH, 2019; RAGHAVAN; WONG, 1986).

O desafio tecnológico unifica o título deste TCC: a coleta de dados via *web scraping* (Etapa 1) (KAUR; PRASHAR, 2025; DILMEGANI, 2025a) é a causa da complexidade da análise de PLN (Etapa 2). A coleta de dados do mundo real enfrenta barreiras de engenharia, como conteúdo dinâmico (TORCATO, 2023; CHEN, 2024) e medidas *anti-scraping* (JAYAN, 2025), e constantemente produz um corpus de dados "ruidoso" e "não estruturado" (BHUVA, 2025). Este ruído caracterizado por sinônimos, abreviações, erros e dados incompletos é o cenário real onde os algoritmos de correspondência devem operar. Portanto, a justificativa tecnológica do trabalho é avaliar empiricamente qual dessas técnicas (léxica, vetorial ou semântica) oferece a melhor combinação de robustez ao ruído para um *dataset* de supermercado real.

Por fim, a execução deste projeto é fundamental para a formação do aluno, pois exige a aplicação e o aprofundamento de conhecimentos em áreas-chave da Ciência da Computação. O trabalho abrange o *pipeline* completo de um projeto de *data science*: desde a engenharia de dados (coleta e estruturação via *web scraping*), passando pelo Processamento de Linguagem Natural (PLN) (JURAFSKY; MARTIN, 2025) e aprendizado de máquina (utilização do modelo pré-treinado BERT), até a análise crítica e avaliação rigorosa de algoritmos e o gerenciamento do ciclo de vida de um projeto de software.

1.2 Objetivos

1.2.1 Objetivo Geral

Analisar comparativamente o desempenho de técnicas clássicas (Distância de Levenshtein, Similaridade Jaccard, Similaridade Jaro-Winkler), vetoriais (TF-IDF com Similaridade de Cossenos) e semânticas (SBERT) para a tarefa de correspondência textual

de produtos de supermercado, utilizando um *dataset* de dados heterogêneo coletado de fontes de *e-commerces* brasileiros.

Adicionalmente, a pesquisa busca verificar a hipótese inicial de que a abordagem semântica (SBERT) ofereceria maior eficácia na resolução de entidades em comparação aos métodos léxicos e vetoriais tradicionais. Parte-se da premissa de que a capacidade dos modelos baseados em *Transformers* de capturar o contexto e o significado dos termos — superando barreiras de sinonímia e abreviações não padronizadas — resultaria em um desempenho superior (*F1-Score*) no cenário desafiador de descrições de produtos ruidosas e não estruturadas.

1.2.2 Objetivos Específicos

1. Realizar um levantamento bibliográfico sobre técnicas de *web scraping*, pré-processamento de texto, PLN e algoritmos de similaridade/correspondência aplicados à resolução de entidades, com foco em produtos de *e-commerce*.
2. Identificar e selecionar um conjunto representativo de *websites* de *e-commerce* como fontes de dados.
3. Desenvolver e implementar *scripts* de *web scraping* para coletar dados relevantes dos produtos (título, descrição, especificações, preço, etc.) dos sites selecionados.
4. Implementar rotinas para limpeza, normalização e pré-processamento dos dados textuais coletados, seguindo práticas padrão de PLN.
5. Selecionar e implementar um conjunto de técnicas e algoritmos de correspondência de produtos baseados em texto (ex: métricas de similaridade de strings, TF-IDF com similaridade de cossenos, etc.).
6. Definir métricas de avaliação de desempenho.
7. Executar experimentos aplicando as técnicas implementadas ao conjunto de dados coletados.
8. Analisar e comparar quantitativamente o desempenho das diferentes abordagens de correspondência.
9. Documentar todo o processo, incluindo os desafios encontrados, as soluções adotadas, os resultados obtidos e as conclusões sobre a viabilidade e eficácia das técnicas estudadas.

2 Fundamentação Teórica

Nesta seção, são apresentados os conceitos teóricos fundamentais que sustentam o desenvolvimento deste trabalho, abrangendo as tecnologias e metodologias empregadas para a coleta, tratamento, análise e gerenciamento dos dados e do projeto.

2.1 Web Scraping

A viabilidade de projetos de ciência de dados e aprendizado de máquina é intrinsecamente dependente da disponibilidade de dados de qualidade. Na ausência de conjuntos de dados curados ou APIs (*Application Programming Interfaces*) que forneçam acesso estruturado, o web scraping emerge como a principal metodologia para a aquisição de dados em larga escala diretamente de fontes públicas na *internet*.

2.1.1 Definição, Processo e Aplicações no Comércio Eletrônico

O *web scraping* (ou extração de dados da web) é formalmente definido como o processo automatizado de coleta e extração de informações de *websites* (MIM, 2021). Este processo utiliza *bots*, ou *scrapers*, que simulam a navegação humana ou requisitam diretamente o conteúdo de um servidor, analisando o código-fonte da página – tipicamente HTML, CSS e JavaScript – para extrair dados específicos e exportá-los para um formato estruturado, como CSV, JSON, ou um banco de dados.

O processo metodológico do *web scraping* envolve etapas bem definidas:

- **Identificação do(s) *websites***: Definição das fontes de dados (neste TCC, quatro fontes de supermercados).
- **Análise da Estrutura**: Inspeção do *Document Object Model* (DOM) do *website* para identificar os seletores (ex: tags HTML, classes CSS) que encapsulam os dados de interesse (ex: nome do produto, preço).
- **Desenvolvimento do *Script***: Codificação do *scraper* utilizando bibliotecas e ferramentas específicas para requisitar a página e aplicar os seletores.
- **Execução e Coleta**: Operação do *script* para iterar sobre as páginas e extrair os dados.
- **Armazenamento e Limpeza**: Persistência dos dados brutos e aplicação de uma limpeza inicial.

No domínio do *e-commerce*, o *web scraping* é uma ferramenta estratégica fundamental. Empresas utilizam a técnica para monitoramento de preços de concorrentes, análise de níveis de estoque, agregação de avaliações de produtos e tendências de mercado (MIM, 2021). A coleta de dados de supermercados, foco deste trabalho, insere-se diretamente nesta aplicação (Actowiz Solutions, 2024).

2.1.2 Desafios Técnicos: *Websites* Estáticos vs. Conteúdo Dinâmico

A complexidade de um projeto de *web scraping* é ditada primariamente pela arquitetura do *website-alvo* (STSIOPKINA, 2024). *Websites* estáticos, que servem todo o conteúdo diretamente no arquivo HTML inicial, podem ser analisados de forma eficiente por *parsers* simples, como o BeautifulSoup[1] (GUAN, 2024).

Contudo, o comércio eletrônico moderno é caracterizado por *websites* de alta complexidade e dinamismo (Actowiz Solutions, 2024). Nestes casos, o servidor envia um HTML inicial que atua apenas como um "*shell*" ou *placeholder* (DILMEGANI, 2025b). Os dados reais – listagens de produtos, preços e descrições – não estão presentes no HTML inicial; eles são carregados subsequentemente através de chamadas JavaScript assíncronas (AJAX) (STSIOPKINA, 2024).

Este paradigma torna *parsers* simples, que não executam JavaScript, ineficazes, pois eles coletam apenas os *placeholders* vazios (DILMEGANI, 2025b). A coleta de conteúdo dinâmico exige, portanto, o uso de ferramentas de automação de navegador, como o Selenium[2]. O Selenium permite ao *scraper* controlar um navegador real (ex: Chrome, Firefox), que por sua vez executa todo o JavaScript da página, simula interações humanas (como cliques, preenchimento de formulários ou rolagem de página – infinite scrolling) e aguarda que o conteúdo dinâmico seja renderizado antes de extrair os dados (GUAN, 2024). A escolha desta ferramenta, embora mais lenta e mais intensiva em recursos (STSIOPKINA, 2024), é frequentemente uma necessidade metodológica para lidar com a complexidade dos *websites* de *e-commerce* (Actowiz Solutions, 2024).

2.1.3 Mecanismos de Contenção (*Anti-Scraping*) e Estratégias de Mitigação

A extração de dados é frequentemente vista como antagônica pelos proprietários dos *websites*, que implementam barreiras técnicas e computacionais para distinguir e bloquear o tráfego de bots (DILMEGANI, 2025b). O *web scraping* é, portanto, um "jogo de gato e rato" que exige estratégias adaptativas (GUESDON, 2024).

As principais técnicas de *anti-scraping* incluem (DILMEGANI, 2025b):

[1] Documentação BeautifulSoup - https://beautiful-soup-4.readthedocs.io/en/latest/
[2] Documentação Selenium - https://selenium-python.readthedocs.io/

- **Bloqueio de IP**: A medida mais comum, onde um *website* bane o endereço de IP (*Internet Protocol*) de um *client* que faz um número excessivo de requisições em um curto período (MIM, 2021). A mitigação envolve a rotação de IPs por meio de serviços de *proxy*.
- ***CAPTCHA***: Testes (ex: "Não sou um robô") projetados para serem fáceis para humanos, mas difíceis para bots (DILMEGANI, 2025b).
- **Análise de robots.txt**: Um arquivo de texto na raiz do servidor que sugere (não obriga tecnicamente) aos bots quais diretórios não devem ser acessados. Embora não seja uma barreira, ignorá-lo é uma violação da "etiqueta" da web (DILMEGANI, 2025b).
- **Mudanças na Estrutura do *Website***: Alterações frequentes nos seletores HTML/CSS (ex: mudar o nome de uma classe CSS) são feitas para "quebrar" *scrapers* que dependem de uma estrutura estável (DILMEGANI, 2025b).

2.1.4 Considerações Éticas e Legais no Contexto Brasileiro (LGPD)

A legalidade do *web scraping* permanece uma "área cinzenta" (Iubenda, 2024). A legitimidade da prática depende fundamentalmente de dois eixos: o quê é coletado e como é coletado.

No que tange aos dados, a coleta de informações publicamente disponíveis, como nomes e preços de produtos em um *e-commerce*, é geralmente considerada legal e ética para fins de pesquisa ou análise de mercado (Salishsea Consulting, 2024). A complexidade legal surge ao lidar com dados pessoais. No Brasil, a Lei Geral de Proteção de Dados (LGPD), Lei nº 13.709/2018, impõe regras estritas sobre o tratamento de qualquer informação relacionada a uma pessoa natural identificada ou identificável (ERMAKOVICH, 2025). Ao focar estritamente em dados de produtos não pessoais, este trabalho se posiciona em um terreno legal significativamente mais seguro.

No que tange ao método, a prática pode ser contestada se violar os Termos de Serviço (ToS) do *website*, que frequentemente proíbem a coleta automatizada. O scraping ético, portanto, envolve um conjunto de boas práticas: não sobrecarregar os servidores do *website-alvo* (MIM, 2021), respeitar as diretrizes do robots.txt e limitar a coleta ao estritamente necessário para os fins da pesquisa (PromptCloud, 2025).

2.2 Engenharia e Pré-processamento de Dados Textuais

Os dados obtidos via *web scraping* raramente estão em formato adequado para a análise computacional. Eles são inerentemente "sujos", ruidosos e não estruturados

(PromptCloud, 2025). O pré-processamento textual é, portanto, uma etapa indispensável em qualquer pipeline de Processamento de Linguagem Natural (PLN), sendo fundamental para a qualidade e o desempenho dos modelos subsequentes (DEEPANSHI, 2025).

2.2.1 O Pipeline de Limpeza e Preparação de Textos em PLN

O pré-processamento textual é um processo de engenharia de dados que transforma texto bruto em representações numéricas ou canônicas. Os objetivos primários desta etapa são (DEEPANSHI, 2025):

- **Redução de Ruído**: Eliminar informações irrelevantes que não agregam valor semântico para a tarefa (ex: tags HTML, URLs, pontuação).
- **Redução de Dimensionalidade**: Diminuir a complexidade do vocabulário, tratando diferentes formas de uma mesma palavra (ex: "Produto", "produto", "PRODUTO") como uma única entidade.
- **Extração de *Features***: Preparar o texto para a vetorização (ex: TF-IDF) ou para a entrada em modelos de *deep learning* (ex: BERT).

Este processo é tipicamente executado como um pipeline, onde o texto passa por uma sequência ordenada de transformações (SHENOY, 2022).

2.2.2 Técnicas Fundamentais: Tokenização, Normalização e Remoção de Stopwords

Embora a ordem e a aplicação das etapas possam variar conforme o domínio, um pipeline de pré-processamento padrão em PLN inclui as seguintes técnicas (DEEPANSHI, 2025):

- **Normalização (*Lowercasing*)**: Conversão de todo o texto para caixa baixa. Este é um passo essencial de normalização, pois, para um modelo computacional, "Leite" e "leite" são tokens distintos. A normalização garante que eles sejam tratados como o mesmo item lexical (SHENOY, 2022).
- **Remoção de Pontuação e Caracteres Especiais**: Eliminação de caracteres como . , ! $ () * % @ que, na maioria das tarefas de PLN, adicionam ruído semântico (DEEPANSHI, 2025).
- **Tokenização**: O processo de segmentar o texto contínuo em unidades discretas, chamadas *tokens*. Geralmente, a tokenização divide uma sentença em suas palavras constituintes (DEEPANSHI, 2025).

- *Remoção de Stopwords*: Stopwords são palavras de alta frequência e baixo significado semântico (ex: “de”, “para”, “em”, “um”, “que”). A remoção destas palavras foca a análise nos termos que de fato carregam o significado do texto (DEEPANSHI, 2025).

2.2.3 Desafios Específicos em Descrições de Produtos de Varejo

A aplicação ingênua do *pipeline* de pré-processamento padrão, descrito acima, é frequentemente prejudicial no domínio específico da correspondência de produtos. O desafio central na limpeza de dados de *e-commerce* não é apenas remover o ruído, mas preservar e normalizar informações críticas que se parecem com ruído (KÖPCKE et al., 2012).

Em descrições de produtos, números e pontuações não são ruído, são *features* essenciais. Por exemplo:

- **Unidades de Medida**: Na comparação “COCA-COLA 2L” vs. “COCA-COLA 500ML”, a remoção de números e pontuação (sugerida pela etapa padrão de remoção de pontuação) resultaria em “COCA COLA L” e “COCA COLA ML”. A informação distintiva crucial (2 vs. 500) seria perdida.
- **Especificações Técnicas**: Marcas, modelos, voltagens e calibres são frequentemente expressos com hifens, barras e números (ex: "castanha-do-para", "cap cafe c/ leite").

A literatura aponta desafios similares em domínios especializados, como o médico, onde abreviações não convencionais (ex: “cap” para “capsula”) e a normalização de unidades de medidas exigem regras de pré-processamento customizadas (KÖPCKE et al., 2012).

Portanto, o pré-processamento para este TCC exige uma abordagem customizada: em vez de remover números e unidades, o objetivo deve ser normalizá-los. Por exemplo, “2L”, “2lts” e “2 litros” devem ser canonicamente transformados em “2000ML”, preservando a informação quantitativa de forma consistente para a subsequente etapa de correspondência (KÖPCKE et al., 2012).

2.3 Processamento de Linguagem Natural (PLN) e Modelos de Representação

O Processamento de Linguagem Natural (PLN), ou *Natural Language Processing* (NLP), é o campo da ciência da computação, inteligência artificial e linguística computacional focado na interação entre computadores e a linguagem humana (JURAFSKY; MARTIN, 2025).

2.3.1 Fundamentos e Aplicações do PLN

O PLN fornece as ferramentas teóricas e computacionais para que máquinas possam processar, analisar, "compreender" e gerar linguagem humana. O campo abrange uma vasta gama de tarefas, desde a análise morfológica e sintática até aplicações complexas, como tradução automática, análise de sentimentos, sumarização de textos e sistemas de diálogo (CARVALHO; COSTA, 2023).

Para o contexto deste trabalho, as subáreas mais relevantes do PLN são a Recuperação de Informação (*Information Retrieval*) e a Mineração de Textos (*Text Mining*), que lidam fundamentalmente com a quantificação da similaridade e relevância entre documentos textuais não estruturados (CARVALHO; COSTA, 2023).

2.3.2 A Evolução dos Modelos de Representação Textual

Para que algoritmos de aprendizado de máquina possam processar textos, a linguagem deve ser convertida em uma representação numérica. A evolução do PLN pode ser compreendida através da evolução desses modelos de representação. As seis técnicas comparadas neste TCC não são escolhas aleatórias; elas representam três eras distintas na busca por representações textuais mais fidedignas e semanticamente ricas.

1. **Era Clássica (Baseada em Caracteres/*Tokens*)**: Os primeiros modelos tratam o texto como uma sequência literal de caracteres ou um conjunto de *tokens*. A similaridade é medida por sobreposição (Jaccard) ou distância de edição (Levenshtein, Jaro-Winkler). Esta abordagem é computacionalmente leve, mas semanticamente ingênua.

2. **Era Vetorial (Hipótese Distribucional)**: Esta era é fundamentada na Hipótese Distribucional de Harris (1954), que postula que palavras que ocorrem em contextos similares tendem a ter significados similares. O texto deixa de ser uma *string* e passa a ser um vetor em um espaço de alta dimensão. Modelos como Bag-of-Words (BoW) e Term Frequency-Inverse Document Frequency (TF-IDF) (SILGE; ROBINSON, 2017) transformam textos em vetores baseados na frequência e relevância dos termos.

3. **Era Semântica (Modelos *Transformer*)**: A era atual, impulsionada por arquiteturas de *deep learning* como os *transformers* (ex: BERT). Estes modelos não se baseiam mais em contagens de palavras, mas aprendem representações vetoriais (*embeddings*) profundas e contextuais. Um modelo pré-treinado como o BERT "compreende" que "achocolatado" e "chocolate em pó" são semanticamente próximos, mesmo sem compartilharem tokens (SHENOY, 2022).

Este trabalho posiciona-se exatamente na intersecção destas eras, testando empiricamente a hipótese de que modelos semanticamente mais ricos (Era 3) superam modelos vetoriais (Era 2), que por sua vez superam modelos clássicos (Era 1), na tarefa complexa de correspondência de produtos.

2.4 Resolução de Entidades e Correspondência de Produtos (*Product Matching*)

Esta seção define o problema central do TCC: a identificação de registros que, embora textualmente distintos, referem-se à mesma entidade do mundo real.

2.4.1 O Problema da Resolução de Entidades (*Entity Resolution*)

A Resolução de Entidades (*Entity Resolution*, ER), amplamente conhecida na literatura por sinônimos como *record linkage*, *duplicate record detection* (detecção de duplicatas) ou *merge/purge*, é o processo de identificar e mesclar registros em um ou mais bancos de dados que representam a mesma entidade do mundo real (SINGLA; DOMINGOS, 2006).

Este é um problema clássico e fundamental na integração e limpeza de dados (BRIZAN; TANSEL, 2015). Conforme destacado no influente *survey* de Elmagarmid, Ipeirotis e Verykios (2007), a detecção de duplicatas é um passo custoso, mas essencial, para garantir a qualidade dos dados antes de qualquer processo de mineração ou análise. A dificuldade reside no fato de que os registros duplicados raramente são sintaticamente idênticos, apresentando erros de digitação, abreviações, ou diferentes convenções de formatação (SUBRAMANIY; PANDIAN, 2012).

2.4.2 Desafios da Correspondência de Produtos em Fontes Heterogêneas

A correspondência de produtos (*product matching*), como a realizada neste TCC, é um subproblema de ER notoriamente desafiador (STEFANIDIS et al., 2014). A dificuldade é amplificada pelo fato de os dados serem provenientes de múltiplas fontes de *e-commerce*, que são, por definição, heterogêneas (PAPADAKIS et al., 2016).

A heterogeneidade de dados no *e-commerce* manifesta-se de várias formas:

- **Heterogeneidade de Esquema (Estrutural)**: As fontes de dados (supermercados) não compartilham o mesmo esquema de banco de dados. Um supermercado pode armazenar a marca em um atributo *brand*, enquanto outro a insere no atributo *title*, ou em algum outro atributo qualquer.

- **Heterogeneidade de Formato**: Os dados podem ser estruturados (tabelas), semi-estruturados (JSON, XML) ou texto livre não estruturado (descrições longas) (PAPADAKIS et al., 2020).
- **Heterogeneidade de Representação**: Este é o desafio central do *fuzzy matching*. O mesmo produto do mundo real é descrito textualmente de formas distintas por diferentes varejistas. Por exemplo:
  - Fonte A: "Achocolatado em Pó Nescau 2.0 Lata 200g"
  - Fonte B: "Nescau Achocolatado 200g"
  - Fonte C: "CHOCOLATE PO NESCAU LT 200G"

O problema a ser resolvido por este TCC não é apenas "encontrar duplicatas" (registros idênticos), mas sim integrar e harmonizar catálogos de produtos de fontes distintas e não-cooperativas, que utilizam vocabulários, abreviações e estruturas de dados diferentes para descrever os mesmos itens (STEFANIDIS et al., 2014).

2.4.3 Técnicas de Similaridade Textual e Correspondência (*Fuzzy Matching*)

Para resolver o problema da Resolução de Entidades em dados textuais, é necessário empregar algoritmos de *fuzzy matching* (correspondência aproximada) (RAGKHITWETSAGUL; KRINKE; CLARK, 2017). Estes algoritmos quantificam a "distância" ou "similaridade" entre duas strings de texto. A Tabela 1 apresenta um quadro comparativo das seis metodologias analisadas neste trabalho, agrupadas por seus princípios fundamentais.

Tabela 1 – Quadro Comparativo das Técnicas de Correspondência Textual

Grupo	**Técnica**	**Nível de Análise**	**Princípio de Funcionamento**
Clássico	Distância de Levenshtein	Caractere	Mede a distância de edição (inserção, exclusão, substituição).
Clássico	Coeficiente de Jaccard	Token (Conjunto)	Mede a sobreposição de conjuntos de tokens (palavras)
Clássico	Distância de Jaro-Winkler	Caractere (com Bônus)	Mede a similaridade de caracteres com bônus para prefixos comuns
Vetorial	BoW + Similaridade de Cossenos	Termo (Frequência)	Mede o ângulo entre vetores de contagem de palavras
Vetorial	TF-IDF + Similaridade de Cossenos	Termo (Relevância)	Mede o ângulo entre vetores de frequência ponderada pela raridade
Semântico	SBERT + Similaridade de Cossenos	Semântica (Contexto)	Mede o ângulo entre vetores de *embeddings* contextuais.

Fonte: Elaborado pelo autor (2025).

2.4.3.1 Métricas Clássicas Baseadas em Caracteres e Tokens

Este grupo de métricas opera diretamente na superfície do texto (na *string* de caracteres ou no conjunto de *tokens*). São algoritmos determinísticos, rápidos e amplamente utilizados para capturar similaridades tipográficas (MUKHERJEE, 2025).

2.4.3.1.1 Distância de Levenshtein

A Distância de Levenshtein, também conhecida como *edit distance* (distância de edição), é uma das métricas de *string* mais fundamentais (LEVENSHTEIN, 1966). Ela é definida como o número mínimo de operações de um único caractere (inserção, exclusão ou substituição) necessárias para transformar uma *string* na outra (GUSFIELD, 1997).

Por exemplo, a distância entre "Nescau" e "Nescal" é 1 (uma substituição). Esta métrica é altamente eficaz para capturar erros de digitação (KAUFMAN; KLEVS, 2021). Sua principal limitação é a ingenuidade semântica: a distância entre "Leite Integral UHT" e "Leite Integral em pó" (5 substituições) é menor do que entre "Leite Integral UHT" e "Leite Integral Longa Vida" (10 operações), embora semanticamente o oposto seja verdadeiro.

2.4.3.1.2 Coeficiente de Jaccard

O Coeficiente de Jaccard (ou Índice de Jaccard) é uma métrica baseada em conjuntos (*sets*) (KYSELA, 2018). No contexto textual, as strings são primeiro tokenizadas (divididas em palavras). A similaridade é então calculada como o tamanho da interseção dos dois conjuntos de *tokens* dividido pelo tamanho da união desses conjuntos (DEWILDE, 2024).

A fórmula é definida como:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Esta métrica é robusta à ordem das palavras. Por exemplo, "Leite Longa Vida Integral" e "Leite Integral Longa Vida" teriam uma alta similaridade de Jaccard, enquanto a Distância de Levenshtein as consideraria muito diferentes.

2.4.3.1.3 Distância de Jaro-Winkler

A métrica Jaro-Winkler é uma evolução da Distância de Jaro, desenvolvida especificamente para tarefas de *record linkage* (MALAGA, 2025). A Distância de Jaro calcula a similaridade com base nos caracteres correspondentes e no número de transposições (caracteres correspondentes que estão fora de ordem) (KYSELA, 2018).

A inovação de Winkler (a Distância de Jaro-Winkler) é a adição de um "bônus de prefixo" (JARO, 1989). Esta métrica atribui um peso maior (maior similaridade) a strings que compartilham um prefixo comum (combinam no início), partindo da observação empírica de que erros de digitação são mais raros no começo de nomes ou marcas (MALAGA, 2025). Isso a torna particularmente adequada para comparar nomes de marcas dentro das descrições de produtos.

2.4.3.2 Métricas Vetoriais Baseadas em Frequência

Este grupo de métricas abandona a comparação direta de strings e adota a abordagem do "espaço vetorial". O texto é primeiro transformado em um vetor numérico, e a similaridade é então calculada entre esses vetores.

2.4.3.2.1 *Bag-of-Words* (BoW)

O modelo *Bag-of-Words* (BoW) é a representação vetorial mais simples. O texto é tratado como um "saco de palavras", ignorando completamente a ordem, a gramática e a estrutura da sentença. Um vocabulário de todos os *tokens* únicos do *corpus* é criado. Cada documento é então representado por um vetor do tamanho do vocabulário, onde cada posição contém a frequência (contagem) daquela palavra no documento (KIM; KIM; CHO, 2017). A similaridade entre dois documentos BoW pode ser calculada usando a Similaridade de Cossenos.

2.4.3.2.2 TF-IDF e a Similaridade de Cossenos

Esta é uma técnica combinada que define o padrão-ouro da Recuperação de Informação clássica. Ela consiste em duas componentes: o método de vetorização (TF-IDF) e o método de comparação (Similaridade de Cossenos).

TF-IDF (*Term Frequency-Inverse Document Frequency*) é uma evolução do BoW que substitui a contagem de frequência bruta por um peso estatístico que reflete a importância de um termo em um documento, em relação a um corpus. O peso $W_{i,j}$ do termo i no documento j é o produto de duas medidas:

1. *Term Frequency* ($tf_{i,j}$): A frequência do termo i no documento j.
2. *Inverse Document Frequency* (idf_i): O logaritmo da divisão do número total de documentos pelo número de documentos que contêm o termo i.

O *idf* penaliza palavras comuns (ex: "cafe", "leite") que aparecem em muitos documentos, e aumenta o peso de palavras raras e distintivas (ex: "extraforte", "prestigio") (SILGE; ROBINSON, 2017). O resultado é um vetor que representa a relevância de cada termo para aquele documento específico (DALVI et al., 2024)(GARTNER, 2024).

Já a similaridade de Cossenos (*Cosine Similarity*), uma vez que os documentos são representados como vetores TF-IDF (ou BoW), a similaridade entre eles é calculada. Em vez de usar a distância Euclidiana, a Similaridade de Cossenos é preferida porque ela mede a orientação (o ângulo) entre os vetores, ignorando suas magnitudes (um fator importante, já que descrições de produtos têm comprimentos variados) (SHARMA, 2019).

A fórmula, derivada do produto escalar, é:

$$\text{Similaridade de Cossenos} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \times ||\mathbf{B}||}$$

Onde **A** e **B** são os vetores TF-IDF. Um resultado de 1 (ângulo de 0°) indica que os vetores são idênticos em orientação (máxima similaridade). Um resultado de 0 (ângulo de 90°) indica que são ortogonais (não compartilham termos relevantes) (SHARMA, 2019).

2.4.3.3 Métricas Semânticas Baseadas em Transformers

A principal limitação dos métodos vetoriais (BoW e TF-IDF) é que eles são puramente lexicais. Eles não possuem noção de semântica. Se duas strings não compartilham *tokens*, sua similaridade será zero, mesmo que sejam sinônimos (ex: "achocolatado" vs. "chocolate em pó").

2.4.3.3.1 Sentence-BERT (SBERT)

Para capturar a similaridade semântica, são necessários modelos de linguagem pré-treinados, como o BERT (*Bidirectional Encoder Representations from Transformers*) (DEVLIN et al., 2019). O BERT é um modelo que aprende representações contextuais profundas da linguagem (MARTINS et al., 2025).

Contudo, o BERT (e modelos similares) não foi projetado para tarefas de similaridade em larga escala. Ele utiliza uma arquitetura cross-encoder, onde um par de sentenças deve ser passado junto ao modelo para obter um escore de similaridade. Conforme apontado por Reimers e Gurevych (2019), encontrar o par mais similar em 10.000 sentenças exigiria 50 milhões de inferências, um processo que seria muito custoso.

O Sentence-BERT (SBERT) foi proposto para resolver essa ineficiência computacional. SBERT modifica o BERT usando uma arquitetura de rede Siamesa (*Siamese network*). Em vez de processar pares, o SBERT processa cada sentença (ou descrição de produto) individualmente e gera um vetor de tamanho fixo (*embedding*) (REIMERS; GUREVYCH, 2019).

O SBERT é treinado para mapear sentenças semanticamente similares para pontos próximos no espaço vetorial (REIMERS; GUREVYCH, 2019). O resultado é um *embedding* que captura o significado semântico da descrição do produto.

Com esta arquitetura, a tarefa de correspondência de produtos se torna computacionalmente eficiente:

- Cada descrição de produto no corpus é passada pelo SBERT uma vez para gerar seu vetor semântico.

- A similaridade entre dois produtos é calculada usando a Similaridade de Cossenos sobre esses vetores.

Este processo reduz o tempo de busca de forma considerável (REIMERS; GUREVYCH, 2019), permitindo que a similaridade semântica seja usada para comparar milhares de produtos eficientemente.

2.5 Métricas de Avaliação de Desempenho

A avaliação de sistemas de Recuperação da Informação (RI) e de aprendizado de máquina requer métricas que quantifiquem a eficácia do modelo em classificar corretamente os dados. Em cenários de correspondência de produtos (*product matching*) e Resolução de Entidades, onde a quantidade de pares "não correspondentes" supera vastamente a de pares "correspondentes", sendo assim, métricas simples como a Acurácia (*Accuracy*) podem ser enganosas (CHRISTEN, 2012). Portanto, a literatura consolidada, como Manning, Raghavan e Schütze (2008), recomenda o uso de métricas baseadas na Matriz de Confusão: Precisão, Revocação (*Recall*) e *F1-Score*.

2.5.1 Matriz de Confusão

A base para o cálculo das métricas é a Matriz de Confusão, uma tabela que cruza as predições do modelo com a realidade dos dados (ground truth). Em uma classificação binária (são produtos idênticos, ou não), existem quatro resultados possíveis para cada instância (CHRISTEN, 2012):

- **Verdadeiros Positivos (VP / TP)**: O modelo previu corretamente que os produtos são iguais.
- **Falsos Positivos (FP / FP)**: O modelo previu incorretamente que os produtos são iguais (erro de "alarme falso").
- **Falsos Negativos (FN / FN)**: O modelo falhou em identificar que os produtos eram iguais (erro de omissão).
- **Verdadeiros Negativos (VN / TN)**: O modelo previu corretamente que os produtos são diferentes.

2.5.2 Precisão (*Precision*)

A Precisão mensura a qualidade das predições positivas do sistema. Ela responde à pergunta: "De todos os pares que o modelo classificou como idênticos, quantos realmente

são?" (MANNING; RAGHAVAN; SCHÜTZE, 2008). Uma alta precisão significa que o sistema gera poucos falsos positivos, ou seja, o usuário pode confiar que os produtos apresentados como iguais são, de fato, iguais. A fórmula é dada por:

$$\text{Precisão} = \frac{VP}{VP + FP}$$

2.5.3 Revocação (*Recall*)

A Revocação (ou Sensibilidade) avalia a capacidade do modelo de encontrar todas as instâncias relevantes. Ela responde à pergunta: "De todos os pares idênticos que existem na base de dados, quantos o modelo conseguiu encontrar?" (MANNING; RAGHAVAN; SCHÜTZE, 2008). Em aplicações críticas, como a detecção de fraudes ou diagnósticos médicos, a revocação é frequentemente priorizada para evitar a omissão de casos positivos (CHRISTEN, 2012). A fórmula é dada por:

$$\text{Revocação} = \frac{VP}{VP + FN}$$

2.5.4 *F1-Score*

Existe um *trade-off* natural entre Precisão e Revocação: aumentar a rigidez do modelo tende a aumentar a Precisão, mas diminuir a Revocação (omite mais casos). Para obter uma visão única do desempenho, utiliza-se o *F1-Score*, que é a média harmônica entre as duas métricas (MANNING; RAGHAVAN; SCHÜTZE, 2008).

A média harmônica é preferida à média aritmética porque ela penaliza valores extremos. Para ter um *F1-Score* alto, o modelo precisa ter bom desempenho tanto em Precisão quanto em Revocação simultaneamente. Conforme discutido por Hand e Christen (2017), o *F1-Score* é a métrica padrão para avaliar algoritmos de *linkage* de registros. A fórmula é:

$$F1 = 2 \cdot \frac{\text{Precisão} \cdot \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

2.6 Trabalhos Relacionados

Nesta seção são apresentados trabalhos acadêmicos e softwares disponíveis no mercado que apresentam funcionalidades semelhantes a proposta desse trabalho. O espectro de soluções varia desde ecossistemas comerciais em larga escala até *frameworks* de pesquisa SOTA (*State of the Art*).

Ecossistemas comerciais como o Google Shopping[3] representam uma abordagem baseada em estruturação e padronização. O Google não depende primariamente de *matching* semântico de texto livre, mas sim da imposição de dados. O processo exige que os lojistas enviem feeds de produtos ao *Google Merchant Center* (GMC) e forneçam, sempre que possível, identificadores únicos globais como GTINs (códigos de barras EAN/UPC) e MPNs (Números de Peça do Fabricante) (STRINGER, 2025). O ponto forte dessa abordagem é a alta precisão determinística quando os IDs estão presentes. Porém, seu ponto fraco é a dependência severa desses IDs. Em domínios onde eles não existem (ex: produtos genéricos) ou não são fornecidos, o sistema falha. A lacuna que este TCC aborda é exatamente esse cenário: o *matching* semântico na ausência de identificadores únicos, dependendo apenas de dados textuais "sujos" (SPERBER, 2023).

Similarmente, marketplaces como a Amazon[4] enfrentam o desafio de agregar produtos de milhões de vendedores terceirizados em um único catálogo coeso (JIANG; CAI, 2024). O objetivo da Amazon é a deduplicação: garantir que cada produto corresponda a um único ASIN (*Amazon Standard Identification Number*). Para isso, ela emprega uma abordagem híbrida: primeiramente, exige IDs de produto (GTINs) para forçar o *matching* com ASINs existentes; em segundo lugar, utiliza modelos semânticos proprietários de *deep learning* para inferir relações e garantir a qualidade do catálogo. Assim como o Google, essa é uma abordagem "caixa-preta" e dependente de IDs. A lacuna que este TCC preenche é fornecer uma análise pública e transparente das classes de algoritmos (clássico, vetorial, semântico) aplicadas a um corpus externo (supermercados) onde os IDs não são a fonte primária de correspondência (WALDFOGEL, 2024).

No contexto de aplicações práticas de *web scraping* para monitoramento de preços em nichos específicos, Maciente (2025) desenvolveu uma solução voltada para o mercado de itens virtuais (*skins*) do jogo *Counter-Strike*. O trabalho destaca-se pela implementação de um mecanismo de "raspagem dinâmica", permitindo que o usuário personalize seletores HTML para adaptar a coleta a diferentes layouts de sites. Embora focado em um domínio diferente do varejo alimentar, Maciente (2025) enfrenta e soluciona desafios técnicos similares aos desta pesquisa, como a superação de mecanismos *anti-bot* e a renderização de conteúdo dinâmico via Selenium, validando a necessidade de abordagens robustas na etapa de engenharia de dados.

Já na esfera de plataformas genéricas de comparação de preços, Souza (2025) propôs um protótipo baseado em arquitetura de microsserviços capaz de indexar produtos de qualquer loja que adira ao padrão semântico *Schema.org*. O trabalho é particularmente relevante para esta pesquisa por utilizar algoritmos de similaridade textual, especificamente TF-IDF e Similaridade de Cossenos, para a classificação e validação de produtos na base

[3] Google Shopping - https://www.google.com/shopping?hl=pt-BR
[4] Amazon - https://www.amazon.com.br/

de dados (ontologia). Souza (2025) demonstra a viabilidade do uso de métricas vetoriais para a identificação de produtos em larga escala, corroborando a escolha dessas técnicas como base comparativa neste estudo.

No âmbito acadêmico, o framework JedAI (PAPADAKIS et al., 2017; PAPADAKIS et al., 2020) é um *toolkit open-source* robusto para *benchmarking* de *pipelines* de ER. Seu ponto forte é a flexibilidade em comparar métodos tradicionais; seu ponto fraco é que seus módulos de *matching* são amplamente baseados em métodos estatísticos e sintáticos (ex: Jaccard, TF-IDF). Dessa forma, este TCC vai analisar e comparar se a precisão semântica é superior à dos módulos-padrão do JedAI ao incluir o SBERT e substituir a complexa engenharia de *pipeline* do JedAI por uma *feature* aprendida (o *embedding* semântico).

Finalmente, na vanguarda da pesquisa em *deep learning* está o modelo Ditto (LI et al., 2020), um sistema SOTA que trata a ER como um problema de classificação de pares de sequências usando PLMs (como BERT). O Ditto alcança desempenho SOTA em dados textuais "sujos". No entanto, sua principal limitação é a dependência de *fine-tuning*: ele exige uma quantidade significativa de dados de treinamento rotulados (PEETERS; BIZER, 2023). Este trabalho compara técnicas que não exigem *datasets* de *fine-tuning* rotulados, tendo isso como um favorável quando comprado ao Ditto, principalmente por utilizar uma aplicabilidade prática (SBERT *zero-shot* ou técnicas clássicas).

3 Materiais e Método

A seguir são destacados a metodologia, ferramentas e técnicas a serem tomadas para a realização integral das especificações desta proposta de TCC, seguindo metodologias ágeis para estruturação, organização e melhoria do processo de desenvolvimento.

3.1 Materiais

A natureza desta pesquisa é computacional, tornando a especificação do ambiente de *hardware* e *software* um componente crítico para a rastreabilidade e replicação dos experimentos. Os recursos utilizados são detalhados a seguir:

3.1.1 Hardware

Abaixo são listadas as especificações técnicas da máquina utilizada para a realização deste trabalho:

- **Sistema Operacional**: Linux Mint 21.3[1] "Virginia", versão do Kernel 5.15.0-161-generic.
- **Processador (CPU)**: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz.
- **Memória RAM**: 16GB em dual channel (2x8GB). Esta especificação foi relevante para o processamento de grandes volumes de dados e o carregamento de modelos de *deep learning* como o SBERT (REIMERS; GUREVYCH, 2019), que são tarefas intensivas em memória.

3.1.2 Ambiente e Ferramentas de Desenvolvimento

Abaixo são listadas a linguagem de programação e as ferramentas utilizadas que compõem o ambiente de desenvolvimento deste projeto:

- **Linguagem de Programação**: Python 3.12[2]. Escolhida por sua vasta adoção na comunidade de ciência de dados e seu robusto ecossistema de bibliotecas para PLN.
- **Ambiente de Desenvolvimento Integrado (IDE)**: Pycharm. Utilizado para toda a codificação dos scripts de coleta, limpeza e análise.

[1] Linux Mint 21.3 - https://linuxmint.com/edition.php?id=311
[2] Documentação Python 3.12 - https://www.python.org/downloads/release/python-3120/

3.1.3 Plataformas de Suporte

Também foram utilizadas algumas plataformas de suporte para o desenvolvimento e acompanhamento do projeto, a seguir, segue a lista de plataformas utilizadas:

- **Supabase**[3]: Plataforma *Backend as a Service* (BaaS) utilizada para hospedar o banco de dados PostgreSQL. Armazenou o dataset coletado e facilitou o acesso aos dados via API.
- **GitHub**[4]: Plataforma de hospedagem de código utilizada para o versionamento de todo o código-fonte do projeto, seguindo práticas de transparência científica.
- **Trello**[5]: Ferramenta de gestão de projetos utilizada para a organização das tarefas do TCC no formato Kanban.
- **Obsidian**[6]: Aplicação utilizada para anotações de pesquisa, documentação, diagramas de arquitetura e banco de dados, e gerenciamento do quadro Kanban.

3.1.4 Bibliotecas (*Frameworks* e Pacotes Python)

Abaixo são listadas todas as bibliotecas de Python que foram necessárias para realização dos algoritmos, incluindo as três etapas principais (coleta de dados, limpeza e pré-processamento e aplicação das técnicas de correspondência textual):

- *playwright*[7]: biblioteca de automação de navegador utilizada para a extração de dados (*scraping*) dos quatro *websites* de supermercados.
- *pandas*[8]: biblioteca fundamental para a manipulação, estruturação e limpeza inicial dos dados (MCKINNEY, 2011).
- *openai*[9]: utilizada para acessar a API da OpenAI (modelo *gpt-4o-mini*) para realizar a identificação de marcas nos produtos.
- *python-dotenv*[10]: utilizada para carregar chaves de API e credenciais de banco de dados de forma segura.
- *python-Levenshtein*[11]: implementação da técnica clássica de Distância de Levenshtein.

[3] Supabase - https://supabase.com/
[4] GitHub - https://github.com/
[5] Trello - https://trello.com/
[6] Obsidian - https://obsidian.md/
[7] *Playwright* - https://playwright.dev/
[8] Documentação Pandas - https://pandas.pydata.org/docs/
[9] Documentação OpenAPI - https://platform.openai.com/docs/overview
[10] Documentação Dotenv Python - https://pypi.org/project/python-dotenv/
[11] Documentação *python-Levenshtein* - https://pypi.org/project/python-Levenshtein/

- *jellyfish*[12]: implementação da técnica clássica de Similaridade de Jaro-Winkler.
- *scikit-learn*[13]: biblioteca central de aprendizado de máquina (PEDREGOSA et al., 2011). Utilizada para:
  - Vetorização de texto (BoW com *CountVectorizer* e TF-IDF com *TfidfVectorizer*).
  - Cálculo da Similaridade de Cossenos.
- *sentence-transformers*[14]: biblioteca utilizada para carregar e aplicar o modelo semântico SBERT para gerar *embeddings* (vetores) das descrições dos produtos. Juntamente ao SBERT, foi utilizado especificamente o modelo *all-MiniLM-L6-v2*, responsável por mapear sentenças e parágrafos para um espaço vetorial denso de 384 dimensões, podendo ser usado para tarefas como agrupamento ou busca semântica (Hugging Face, 2024).
- *torch*[15]: framework de *deep learning* que serve como base para a biblioteca *sentence-transformers*.
- *SQLAlchemy*[16]: Kit de ferramentas SQL e Mapeador Objeto-Relacional (ORM) usado para interagir com o banco de dados Postgres.
- *psycopg2-binary*[17]: Adaptador de banco de dados (*driver*) que permite a conexão do Python com o PostgreSQL (Supabase).
- *numpy*[18]: biblioteca de computação numérica, dependência principal do *pandas* e *scikit-learn* para operações com vetores e matrizes.
- *scipy*[19]: biblioteca de computação científica, utilizada para operações estatísticas e de otimização subjacentes.
- *networkx*[20]: biblioteca utilizada para construção de grafos dos produtos similares.
- *matplotlib*[21]: biblioteca-base para a plotagem de gráficos e visualizações da análise de resultados.

[12] Documentação *jellyfish* - https://pypi.org/project/jellyfish/
[13] Documentação scikit-learn - https://scikit-learn.org/stable/
[14] Documentação sentence-transformers - https://sbert.net/
[15] Documentação PyTorch - https://pytorch.org/
[16] Documentação SQLAlchemy - https://docs.sqlalchemy.org/en/20/
[17] Documentação psycopg2-binary - https://pypi.org/project/psycopg2-binary/
[18] Documentação NumPy - https://numpy.org/doc/stable/
[19] Documentação SciPy - https://docs.scipy.org/doc/scipy/
[20] Documentação NetworkX - https://networkx.org/en/
[21] Matplotlib - https://matplotlib.org/

3.2 Método

Este trabalho segue uma metodologia de pesquisa quantitativa, apresentando seus resultados a partir das métricas de precisão, revocação e do *F1-Score*.

A condução de um projeto de ciência de dados, que envolve fases interdependentes de exploração, coleta, implementação e análise, requer uma metodologia de gerenciamento estruturada, mas flexível.

3.2.1 Gerenciamento Ágil com Kanban

Adotou-se o método ágil Kanban para a gestão do projeto. O Kanban é uma metodologia visual que permite o gerenciamento do fluxo de trabalho, focando na visualização de tarefas, limitação do trabalho em progresso (WIP) e melhoria contínua, facilitando a transparência e a adaptabilidade (AHMED et al., 2024). A literatura recente identifica o Kanban como uma metodologia particularmente eficaz para gerenciar os desafios inerentes aos projetos de Ciência de Dados, os quais são caracteristicamente exploratórios e iterativos, diferindo dos ciclos de desenvolvimento de software tradicionais (DAHULE, 2023).

O fluxo de trabalho foi visualizado em um quadro (utilizando as ferramentas Trello e Obsidian) dividido em quatro colunas principais, representando os estágios de cada tarefa:

1. ***Backlog***: Repositório de todas as tarefas a serem executadas.
2. **Em Desenvolvimento**: Tarefas ativas que estavam sendo executadas.
3. **Em Testes**: Tarefas concluídas aguardando validação, debug ou revisão.
4. **Concluído**: Tarefas finalizadas e validadas.

O escopo total da pesquisa foi decomposto em seis "épicos" (macro-entregas), que por sua vez foram subdivididos em aproximadamente 30 tarefas menores:

1. Análise inicial e estudo da aérea de *Product Matching*;
2. *Web Scraping* e Preparação dos Dados;
3. Implementação das Técnicas de Correspondência Textual;
4. Análise dos Resultados e Métricas;
5. Escrita da Monografia;
6. Ajustes Finais e Apresentação.

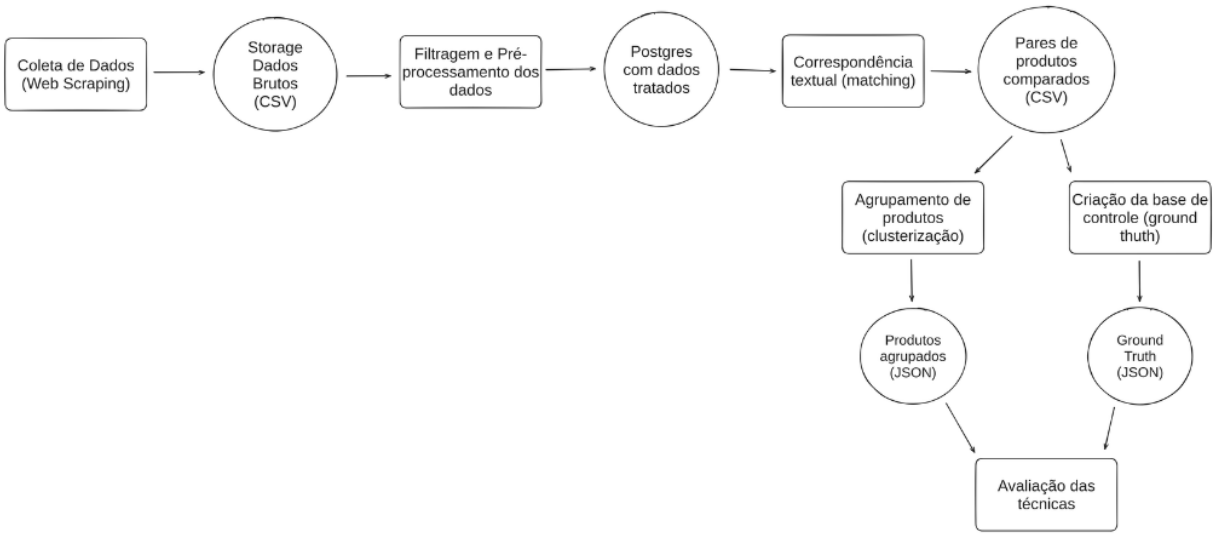

Figura 1 – Diagrama da Arquitetura Geral do pipeline

Fonte: Elaborada pelo autor.

3.2.2 Arquitetura geral adotada

Para operacionalizar a pesquisa, foi desenhada uma arquitetura de *pipeline* de dados linear e modular. A Figura 1 apresenta a visão geral deste fluxo, detalhando a transformação dos dados desde a sua captura bruta até a geração das métricas de desempenho.

O fluxo ilustrado na Figura 1 compreende as seguintes etapas sequenciais:

1. **Aplicação de Coleta**: Execução dos *scripts* de *web scraping* para extração dos dados nas fontes selecionadas.
2. **Armazenamento Bruto**: Persistência inicial dos dados em formato CSV.
3. **Pré-processamento e Filtragem**: Aplicação das rotinas de limpeza e seleção dos produtos da cesta básica.
4. **Banco de Dados (PostgreSQL)**: Armazenamento estruturado dos dados normalizados.
5. **Aplicação das Técnicas**: Processamento dos pares de produtos utilizando as seis abordagens de similaridade (Léxicas, Vetoriais e Semânticas).
6. **Persistência de Pares**: Geração de arquivo CSV contendo os *scores* de similaridade para todos os pares comparados.
7. **Agrupamento**: Algoritmo de clusterização baseado em grafos e limiares de corte.

8. **Produtos Agrupados (JSON)**: Saída estruturada contendo os *clusters* formados por cada técnica.

9. **Geração do Ground Truth**: Algoritmo de consenso para identificar correspondências verdadeiras validadas por múltiplas técnicas.

10. **Base de Controle (JSON)**: *Dataset* final de referência (*Silver Standard*) para validação.

11. **Avaliação**: Cálculo das métricas de Precisão, Revocação e *F1-Score* e geração dos gráficos comparativos.

3.2.3 Coleta e Tratamento de Dados

Este pilar metodológico detalha o *pipeline* de engenharia de dados, desde a aquisição dos dados brutos (fonte primária) até sua transformação em um *dataset* limpo e estruturado, pronto para a modelagem.

3.2.3.1 Coleta de Dados Primários via *Web Scraping*

O *dataset* utilizado nesta pesquisa é de autoria própria, constituindo-se como fonte primária. Os dados (descrições de produtos, preços e categorias) foram coletados de quatro fontes distintas (*websites* de supermercados) através da técnica de *Web Scraping*.

A seleção da biblioteca *Playwright* foi uma decisão técnica deliberada. Diferentemente de analisadores HTML estáticos (como *Beautiful Soup*), o *Playwright* é uma ferramenta moderna de automação de navegador. Esta capacidade é essencial para interagir com *websites* de *e-commerce*, que frequentemente renderizam seu conteúdo dinamicamente via *JavaScript* (ex: carregamento assíncrono de produtos). Ferramentas tradicionais falhariam em extrair dados de tais páginas. A literatura valida o *Playwright* como uma ferramenta robusta para testes end-to-end e extração de dados complexos (DUSEK, 2024).

Durante a coleta, foram observadas as considerações éticas e legais. O processo foi configurado para utilizar taxas de requisição moderadas, respeitando os termos de serviço dos *websites-alvo*, e coletando apenas dados públicos e não sensíveis, com o único propósito de pesquisa acadêmica.

Como resultado, foi obtido um *dataset*, que ainda iria passar por tratamento de limpeza e pré-processamento dos dados coletados. Os dados foram salvos em um arquivo CSV com as seguintes colunas: nome do produto, preço, categoria, código interno (identificador único na plataforma de origem, caso houver), plataforma (*website* de origem) e link de acesso.

3.2.3.2 Pré-processamento e Limpeza de Dados Textuais: Abordagem Clássica

Dados textuais brutos extraídos da web são inerentemente ruidosos, contendo inconsistências, erros de digitação e formatação variada. O pré-processamento é uma etapa crítica em qualquer projeto de PLN, pois a qualidade dos dados afeta diretamente o desempenho dos modelos de análise subsequentes (KATHURIA; GUPTA; SINGLA, 2021).

Nesta pesquisa, foi aplicado um pipeline de normalização de texto padrão em PLN, conforme descrito na literatura, implementado com a biblioteca pandas (MCKINNEY, 2011). As etapas incluíram:

1. *Conversão para Minúsculas* ***Lowercasing***: Conversão de todo o texto para caracteres minúsculos, eliminando a diferenciação por capitalização (ex: “Leite” vs. “leite”).

2. **Remoção de Diacríticos**: Acentuações e outros sinais diacríticos (ex: ´ ‘ ˆ ~ ¨ ç) que podem influenciar negativamente, foram removidos.

3. **Remoção de Pontuação e Símbolos**: Caracteres não alfanuméricos (ex: .,*()$) que não agregam valor semântico para a tarefa de correspondência, foram removidos.

4. **Remoção de Espaços Extras**: espaços em branco múltiplos para um único espaço e espaços em branco no início e no fim da sentença foram removidos.

5. **Extração e Padronização de Unidade de Medidas**: Identificação da unidade de medida, seja ela uma unidade de volume, massa ou comprimento. Seguido da remoção e normalização para mililitros, gramas ou centímetros.

6. **Identificação de Marcas**: Foi utilizado duas abordagens para identificação da marca dos produtos:

   - **Abordagem Baseada em Lista**: Foi criada uma lista das maiores e mais conhecidas marcas, e através desta lista, foi feito uma busca por essas marcas no nome de cada produto.
   - **Abordagem Baseada em Inteligência Artificial Generativa**: Para marcas pouco conhecidas, ou não contidas na lista de marcas, utilizou-se para identificação a API da OpenAI através da biblioteca *openai*, especificamente o modelo gpt-4o-mini. Este modelo foi selecionado por apresentar um balanço ideal entre custo e performance. A tabela Tabela 2 mostra uma pequena amostra dos nomes dos produtos e a marca identificada pelo modelo.

   Após a identificação, foi feita a remoção das marcas dos produtos, para evitar que comparações sejam feitas por marcas, uma vez que é necessário comparar produtos, independente de sua marca.

Tabela 2 – Amostra das marcas identificadas através do nome do produto pelo modelo gpt-4o-mini

Produto	**Marca identificada**
sabonete liquido johnson's baby glicerina camomila 400ml	johnson's
macarrao instantaneo nissin 500g	nissin
gelatina zero acucar sabor framboesa dr. oetker 12g	dr. oetker
leite de coco culinario sococo 200ml	sococo
feijao preto tipo 1 broto legal 1 kg	broto legal
cafe em po bom jesus 500g	bom jesus
oleo de soja vitaliv garrafa 900 ml	vitaliv
leite condensado semidesnatado triangulo mineiro caixa 395g	triangulo mineiro
farinha de arroz aminna 300g	aminna
pao italiano carrefour 500g	carrefour
pao de queijo congelado forno de minas especial 30 anos 400 g	forno de minas

Fonte: Elaborado pelo autor (2025).

Como resultado, foi obtido um *dataset* limpo, organizado e normalizado, pronto para utilização na próxima etapa de aplicação das técnicas de correspondência textual. Os dados tratados foram salvos em um banco de dados Postgres no Supabase.

3.2.3.3 Implementação e aplicação das Técnicas de Correspondência

O cerne desta pesquisa reside na análise comparativa de seis técnicas de correspondência textual. A Tabela 3 apresenta uma síntese das técnicas implementadas e as bibliotecas de software utilizadas para sua aplicação.

Tabela 3 – Síntese das Técnicas Implementadas e Bibliotecas Utilizadas

Técnica	**Biblioteca Utilizada**
Distância de *Levenshtein*	*python-Levenshtein*
Índice de *Jaccard*	Implementação customizada
Similaridade de *Jaro-Winkler*	*jellyfish*
Bag-of-Words (BoW) + Sim. Cossenos	*scikit-learn*
TF-IDF + Sim. de Cossenos	*scikit-learn*
SBERT (*Sentence-BERT*)	*sentence-transformers*, *torch*

Fonte: Elaborado pelo autor (2025).

Para viabilizar a análise comparativa, foi desenvolvido um *script* de execução que implementa um *pipeline* de processamento. Este *script*[22] processa o *dataset* exaustivamente, comparando produtos por categoria, par a par.

Para cada par de produtos, o *pipeline* calculou o *score* de similaridade utilizando, sequencialmente, todas as seis técnicas avaliadas (*Levenshtein, Jaccard, Jaro-Winkler,*

[22] Disponível no Github - https://github.com/rafaelneto2/product-matching

BoW + Similaridade de Cossenos, TF-IDF + Similaridade de Cossenos, e SBERT). Ao final do processamento, todos os resultados (incluindo os IDs dos produtos e os seis *scores* de similaridade) foram agregados e salvos em um único arquivo CSV. Este arquivo consolidado serviu como entrada bruta para a etapa de validação e análise de métricas.

3.2.4 Método de Análise Comparativa e Métricas de Avaliação

Para comparar cientificamente o desempenho das seis técnicas de correspondência, foi necessário estabelecer métricas de avaliação objetivas.

3.2.4.1 Criação do *Ground Truth*

Uma limitação deste trabalho está na construção da base de verdade (Ground Truth), na tentativa de solucionar este problema, foi criado um *dataset* de validação por meio da implementação de um algoritmo que comparou os grupos de produtos resultantes de cada técnica aplicada, e realizou a intercessão para identificar produtos verdadeiramente similares.

3.2.4.2 Métricas de Desempenho

O desempenho foi avaliado usando métricas padrão de classificação, derivadas da Matriz de Confusão, que tabula os resultados em Verdadeiros Positivos (TP), Falsos Positivos (FP), Verdadeiros Negativos (TN) e Falsos Negativos (FN).

- **Precisão (*Precision*)**: Mede a exatidão das previsões positivas. É a proporção de TP sobre todos os positivos previstos ($Precisão = TP/(TP + FP)$). No contexto do *product matching*, alta precisão significa que, quando o modelo afirma que dois produtos são iguais, ele está quase sempre correto (minimizando Falsos Positivos).
- **Revocação (*Recall*)**: Mede a completude do modelo em encontrar todos os positivos. É a proporção de TP sobre todos os positivos reais ($Revocação = TP/(TP + FN)$). Alta revocação significa que o modelo é capaz de encontrar a maioria dos pares corretos existentes no dataset (minimizando Falsos Negativos)
- **F1-Score**: A seleção da métrica primária de avaliação requer uma análise da natureza do problema. A tarefa de *product matching* é, por definição, um problema de classificação altamente desbalanceado.

A análise visual dos resultados, incluindo gráficos da matrizes de confusão, Precisão, Revocação e *F1-Score*, comparando todas as técnicas, foi gerada usando a biblioteca *matplotlib*, e será apresentada no capítulo de Resultados.

4 Desenvolvimento

Nesta seção são detalhados os passos e decisões tomadas ao decorrer do período de desenvolvimento deste trabalho. São apresentadas as etapas de implementação, iniciando pela coleta de dados que irá compor o *dataset*, após, é a apresentado o fluxo de pré-processamento e estruturação dos dados coletados, por fim, a aplicação das técnicas de correspondência textual a partir do *dataset* já padronizado.

4.1 Coleta de dados

A etapa de coleta de dados constitui a fundação sobre a qual toda a análise comparativa de correspondência textual de produtos é construída. A qualidade, a abrangência e a consistência dos dados brutos são determinantes para o sucesso das fases subsequentes de pré-processamento, análise e aplicação de modelos de aprendizado de máquina. Diante da necessidade de obter um conjunto de dados robusto e representativo do mercado varejista de supermercados, optou-se pelo desenvolvimento de uma solução de software customizada para a extração automatizada de informações de produtos, um processo conhecido como *web scraping*.

4.1.1 Concepção e Arquitetura do Sistema

O ponto de partida para o desenvolvimento da ferramenta foi a identificação de um requisito fundamental: a capacidade de extrair informações de produtos de diversas fontes online, cada qual com sua própria estrutura de *website*, tecnologia de apresentação de conteúdo e *layout*. A heterogeneidade e a natureza dinâmica dessas fontes, que frequentemente utilizam tecnologias modernas de carregamento de conteúdo, tornaram inviável o uso de ferramentas genéricas, demandando a criação de uma aplicação sob medida.

4.1.1.1 Princípios Arquiteturais Adotados

Para nortear o desenvolvimento e assegurar a qualidade e a longevidade do software, foram adotados princípios consolidados da Engenharia de Software. A arquitetura do sistema foi projetada com foco em dois pilares principais: Separação de Responsabilidades (*Separation of Concerns*) e Modularidade.

A Separação de Responsabilidades foi implementada por meio de uma arquitetura em camadas, que isola as diferentes preocupações lógicas do sistema. Cada camada possui uma função bem definida: a orquestração do fluxo de trabalho, a definição dos modelos de dados (domínio), o estabelecimento de contratos (interfaces) e a implementação

concreta do acesso à web e persistência de dados (infraestrutura). Essa separação facilita o desenvolvimento, o teste e, principalmente, a manutenção do sistema, uma vez que alterações em uma camada, como a lógica de extração de um *site* específico, não impactam as demais.

A Modularidade e Extensibilidade são alcançadas através do design que permite a adição de novas fontes de dados (supermercados) com o mínimo de esforço e sem a necessidade de modificar o código central da aplicação. O sistema foi concebido como um *framework* onde cada *website*-alvo é tratado por um componente especializado, ou *scraper*. Para adicionar um novo supermercado à coleta, basta desenvolver um novo *scraper* que adere a um contrato pré-definido, promovendo a reutilização de código e a escalabilidade da solução.

Essa abordagem arquitetônica representa uma decisão estratégica que mitiga diretamente os principais riscos associados a projetos de *web scraping*: a fragilidade diante de mudanças nos *websites*-alvo e a dificuldade de escalar a coleta para novas fontes. Ao isolar a lógica específica de cada site em módulos independentes, o sistema garante que a inevitável alteração na estrutura HTML de um dos supermercados exija a atualização de somente um componente, mantendo o restante do sistema íntegro e funcional. Essa resiliência transforma o que poderia ser um simples conjunto de *scripts* em uma plataforma de coleta de dados robusta e escalável.

4.1.1.2 Arquitetura em Camadas

A estrutura do sistema foi organizada em quatro camadas distintas, conforme ilustrado na Figura 2. Essa organização lógica promove um baixo acoplamento e uma alta coesão entre os componentes do software.

As camadas são definidas como:

1. **Orquestração** (*Orchestration*): Representada pelo ponto de entrada main.py, esta camada é responsável por inicializar o sistema, instanciar os *scrapers*, gerenciar o fluxo de execução principal e consolidar os dados coletados.

2. **Domínio** (*Domain*): Contém as entidades de negócio do sistema, ou seja, os modelos de dados que representam os conceitos centrais do problema, como Product, Category e Company.

3. **Interfaces**: Define os contratos formais que os componentes devem seguir. O elemento principal desta camada é a ScraperInterface, que especifica os métodos que todo scraper deve implementar.

4. **Infraestrutura** (*Infrastructure*): Contém as implementações concretas da lógica de extração, acesso à web e persistência de dados. É nesta camada que residem

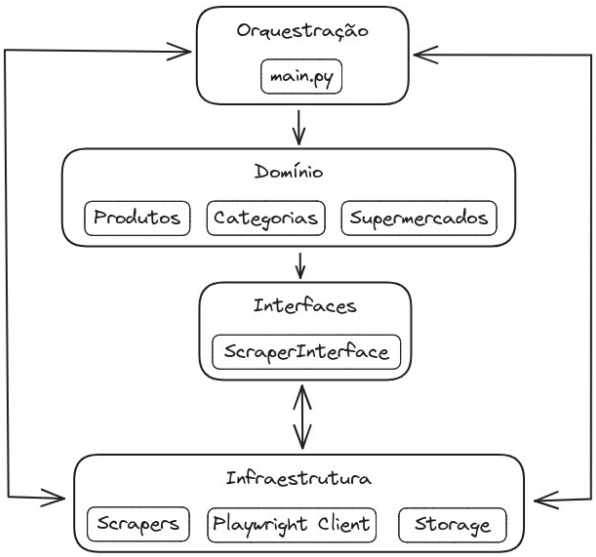

Figura 2 – Diagrama da Arquitetura em Camadas do Sistema de Coleta de Dados

Fonte: Elaborada pelo autor.

os *scrapers* especializados para cada supermercado e os módulos utilitários para interação com a *web* e armazenamento de arquivos.

4.1.2 Fundamentação Tecnológica

A seleção das tecnologias foi um processo criterioso, visando escolher ferramentas que não somente atendessem aos requisitos técnicos do projeto, mas que também representassem o estado da arte para a tarefa de coleta de dados em ambientes *web* modernos e dinâmicos. A pilha tecnológica foi escolhida para enfrentar desafios específicos como o carregamento de conteúdo via JavaScript e a otimização de operações de rede.

4.1.2.1 A Linguagem Python como Base para Coleta e Análise de Dados

A linguagem de programação Python foi selecionada como a espinha dorsal do projeto. A escolha se justifica por múltiplos fatores que a tornam ideal para projetos de ciência de dados. Primeiramente, sua sintaxe clara e intuitiva permite um desenvolvimento mais rápido e um código mais legível, o que é vantajoso em projetos acadêmicos com prazos definidos (Python Software Foundation, 2024).

Mais importante, Python possui um ecossistema de bibliotecas extremamente rico e maduro para manipulação e análise de dados. Ferramentas como Pandas, para manipulação de dados tabulares, e NumPy, para computação numérica, são padrões da indústria (MCKINNEY, 2011; OLIPHANT, 2015) e serão utilizadas nas fases posteriores deste trabalho. A escolha de Python para a etapa de coleta cria, portanto, uma sinergia tecnológica, permitindo que todo o fluxo do projeto, da extração à análise, seja conduzido em um ambiente unificado. Adicionalmente, a vasta e ativa comunidade de desenvolvedores

Python garante amplo suporte, documentação e uma contínua evolução da linguagem e de suas ferramentas.

4.1.2.2 Automação de Navegadores para *Websites* Dinâmicos com Playwright

Um dos maiores desafios técnicos no *web scraping* contemporâneo é lidar com *websites* que carregam seu conteúdo dinamicamente. Muitas plataformas de *e-commerce*, incluindo as dos supermercados-alvo, não enviam a lista completa de produtos na resposta HTML inicial. Em vez disso, utilizam JavaScript para carregar itens à medida que o usuário rola a página (*lazy loading*) ou clica em botões como "Carregar Mais" (*infinite scroll*) (DILMEGANI, 2025b; JONES, 2025). Ferramentas de *scraping* tradicionais, que se baseiam em simples requisições HTTP, são incapazes de executar esse JavaScript e, consequentemente, não conseguem acessar a totalidade dos dados.

Para superar esse obstáculo, foi adotada a biblioteca Playwright[1]. Desenvolvida pela Microsoft, Playwright é uma ferramenta de automação de navegadores que permite controlar programaticamente instâncias reais de navegadores como Chromium, Firefox e WebKit. Ao operar um navegador completo, a ferramenta é capaz de renderizar páginas *web* exatamente como um usuário as veria, executando todo o JavaScript necessário para o carregamento dinâmico do conteúdo (Microsoft, 2025). Suas principais vantagens para este projeto incluem:

- Manipulação de Conteúdo Dinâmico: Playwright possui mecanismos de espera automática que garantem que as interações (como extrair um texto ou clicar em um botão) só ocorram após o elemento correspondente estar visível e disponível na página, eliminando a complexidade de gerenciar manualmente os tempos de carregamento de conteúdo assíncrono (Microsoft, 2025; OK, 2025).
- Simulação de Interação Humana: A biblioteca oferece uma API robusta para simular ações complexas do usuário, como rolar a página, clicar em elementos específicos e preencher formulários, essenciais para navegar pelas páginas de produtos e acionar o carregamento de mais itens.
- Suporte Nativo a Operações Assíncronas: A API do Playwright é projetada para se integrar perfeitamente com o paradigma de programação assíncrona, o que é fundamental para a otimização de desempenho discutida a seguir (SHAMNDY, 2025).

4.1.2.3 Otimização de Desempenho com Operações Assíncronas (*Asyncio*)

A atividade de *web scraping* é, por sua natureza, intensiva em operações de entrada e saída (*I/O-bound*). Em um modelo de execução síncrono tradicional, o programa gasta

[1] Documentação Playwright - https://playwright.dev/docs/intro

a maior parte de seu tempo em estado ocioso, aguardando a resposta da rede para cada requisição enviada a um servidor *web*. Considerando que o sistema precisa acessar múltiplos *sites* e, dentro de cada um, dezenas ou centenas de páginas de categorias, essa ociosidade se torna um gargalo de desempenho significativo, resultando em tempos de coleta excessivamente longos (ANDERSON, 2024).

Para mitigar esse problema, foi utilizada a biblioteca Asyncio[2], parte do ecossistema padrão do Python. O Asyncio permite a escrita de código concorrente utilizando a sintaxe *async/await*. Através de um *event loop*, ele gerencia múltiplas operações de I/O (como requisições de rede) de forma não-bloqueante (GUNTUPALLI, 2022; BOLTON, 2024). Enquanto uma tarefa está aguardando a resposta de um servidor, o *event loop* pode alocar o tempo de processamento para iniciar ou dar continuidade a outras tarefas. Isso possibilita um paralelismo efetivo, onde múltiplas páginas podem ser requisitadas e processadas concorrentemente, eliminando o tempo de espera ocioso e reduzindo drasticamente o tempo total de execução da coleta de dados (KULYK, 2024; CASTILLO, 2024).

A combinação de Python, Playwright e Asyncio constitui uma pilha tecnológica moderna e eficaz. Ela supera as limitações de abordagens mais antigas (como o uso de Requests[3] e BeautifulSoup[4], que falham com conteúdo dinâmico) e oferece uma alternativa de melhor desempenho e com uma API mais limpa em comparação com ferramentas de automação de primeira geração, como o Selenium (SHAMNDY, 2025). A integração nativa entre as capacidades assíncronas do Playwright e o gerenciamento de concorrência do Asyncio cria uma solução poderosa, na qual o Asyncio pode orquestrar tarefas de alto nível (ex: executar *scrapers* de diferentes lojas em paralelo), enquanto o Playwright gerencia eficientemente as interações de baixo nível com o navegador (JOEVU, 2023; PADRóN, 2025). Essa escolha demonstra uma seleção de ferramentas alinhada às melhores práticas atuais da indústria para a resolução do problema em questão.

4.1.3 Fluxo de Execução da Coleta de Dados

A execução do sistema segue um fluxo lógico e sequencial, orquestrado pelo main.py, que combina os componentes descritos para realizar a coleta de dados de ponta a ponta. O processo, ilustrado no fluxograma da Figura 3, é projetado para ser robusto e eficiente.

A descrição detalhada do fluxo de execução ocorre nos seguintes passos:

1. **Inicialização**: A execução é iniciada no main.py. A primeira ação é criar uma lista contendo instâncias de todas as classes de *scraper* concretas disponíveis.

[2] Documentação Asyncio - https://docs.python.org/3/library/asyncio.html
[3] Documentação Requests - https://requests.readthedocs.io/en/latest/
[4] Documentação BeautifulSoup - https://beautiful-soup-4.readthedocs.io/en/latest/

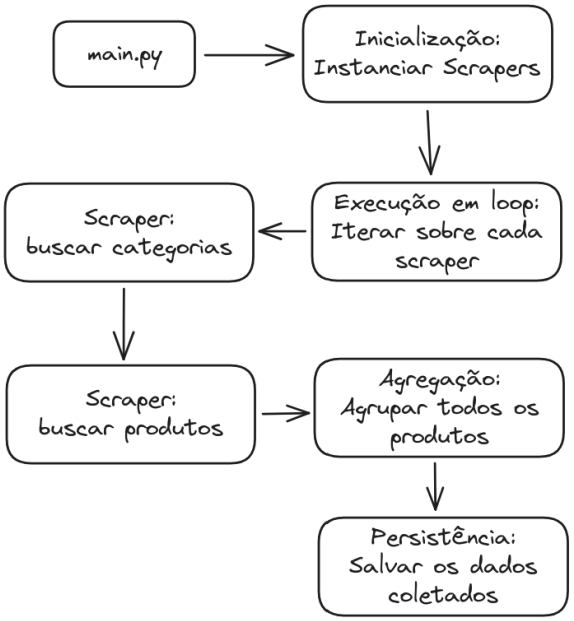

Figura 3 – Fluxograma do Processo de Coleta de Dados

Fonte: Elaborada pelo autor.

2. ***Loop* principal sobre *scrapers***: O sistema inicia um laço de repetição que itera sobre cada objeto *scraper* na lista. Para cada supermercado, o processo de coleta é executado.

3. **Coleta de categorias**: Dentro do laço, o método *scrape_categories()* do *scraper* atual é invocado. Este método é responsável por navegar até a página principal ou mapa do site do supermercado e extrair uma lista de todas as categorias de produtos disponíveis, juntamente com seus respectivos *links* (URLs).

4. ***Loop* sobre categorias**: Com a lista de categorias em mãos, o sistema inicia um segundo laço, aninhado ao primeiro, que itera sobre cada categoria retornada.

5. **Coleta de produtos por categoria**: Para cada categoria, o método *scrape_products* é chamado. O *scraper* então navega para a URL daquela categoria específica.

6. **Interação e extração**: Uma vez na página da categoria, o *scraper* executa a lógica para extrair os dados de cada produto visível. Fundamentalmente, ele também lida com a paginação, seja rolando a página para acionar o *lazy loading* ou clicando em botões "Carregar Mais" até que todos os produtos daquela categoria tenham sido carregados e extraídos. Os dados brutos são então convertidos em objetos Product.

7. **Agregação de dados**: Todos os objetos Product coletados em cada iteração são adicionados a uma única lista mestra, mantida pelo orquestrador.

8. **Persistência**: Após o término do laço principal (ou seja, após todos os *scrapers* terem processado todas as suas categorias), a lista mestra completa é passada para

o componente *CSVWriter*, que a escreve em um arquivo produtos.csv, finalizando o processo de coleta.

Este fluxo de execução em duas fases (primeiro coletar todas as categorias e depois os produtos de cada uma) é uma estratégia de otimização deliberada. Ao criar um "mapa" completo do trabalho a ser feito (a lista de todas as URLs de categorias) antes de iniciar a tarefa intensiva de extração de produtos, o sistema ganha em resiliência e escalabilidade. Se a extração de uma categoria específica falhar, o erro pode ser registrado e o sistema pode continuar para a próxima, sem interromper todo o processo. Além disso, essa estrutura facilita a exploração de maior paralelismo em futuras versões do sistema: uma vez que a lista de categorias é conhecida, múltiplas instâncias de *workers* poderiam, em uma evolução do sistema, processar diferentes categorias simultaneamente, otimizando ainda mais o desempenho. Este *design* reflete um entendimento prático dos desafios inerentes à coleta de dados em larga escala, resultando em um sistema mais robusto, depurável e preparado para futuras melhorias.

4.2 Pré-processamento e Estruturação dos Dados

A etapa de pré-processamento de dados constitui um pilar fundamental, sendo sua execução criteriosa um pré-requisito para a obtenção de resultados robustos e confiáveis (SIINO; TINNIRELLO; CASCIA, 2024). Os dados brutos, especialmente quando oriundos de fontes heterogêneas como o *web scraping*, são caracterizados por uma ausência de padronização, presença de ruídos — como caracteres irrelevantes, inconsistências de formatação e variações lexicais — e uma estrutura inadequada para o consumo direto por algoritmos analíticos (KOUDAS; MARATHE; SRIVASTAVA, 2004). A aplicação de técnicas de correspondência textual sobre dados não tratados resultaria em uma análise de baixa acurácia, cujas conclusões seriam invalidadas pela má qualidade dos insumos.

Diante desse cenário, foi desenvolvido um pipeline de pré-processamento robusto e sequencial, projetado especificamente para transformar o conjunto de dados brutos de produtos de supermercado em um *dataset* estruturado e organizado. Este processo não se resume a uma simples limpeza; ele representa uma etapa metodológica controlada, cujo objetivo é criar um ambiente de dados padronizado e de alta qualidade, essencial para a validade da análise comparativa subsequente. Cada fase do pipeline foi concebida para mitigar um tipo específico de inconsistência, garantindo que os dados alimentados nos modelos de correspondência textual estejam livres de ambiguidades que poderiam comprometer a integridade da pesquisa.

O fluxo de trabalho, detalhado nas subseções seguintes, abrange desde o carregamento inicial dos dados até a sua persistência em um banco de dados relacional. As etapas incluem a limpeza e normalização lexical, a filtragem semântica para otimização

computacional (CHAI, 2022), a extração e padronização de unidades de medida, a identificação híbrida de marcas e, por fim, a remoção destas para isolar o descritivo principal do produto. A Tabela 4 ilustra, por meio de exemplos concretos, o efeito cumulativo dessas transformações, demonstrando a transição de um dado bruto e ruidoso para uma representação final, limpa e estruturada, pronta para a análise.

Tabela 4 – Exemplo da transformação dos dados de produtos no pré-processamento

Dado Bruto	Após Limpeza Lexical	Unidade Extraída	Unidade Normalizada	Marca Identificada	Nome Final do Produto
Açúcar Refinado UNIÃO 1kg	acucar refinado uniao 1kg	1kg	1000	UNIÃO	acucar refinado
Leite Longa Vida Integral Italac 1 L	leite longa vida integral italac 1 l	1 l	1000	ITALAC	leite longa vida integral
Achoc. em Pó NESCAU 2.0 Lata 380g	achoc em po nescau 20 lata 380g	380g	380	NESCAU	achoc em po 20 lata
OLEO DE SOJA LIZA PET 900ML	oleo de soja liza pet 900ml	900ml	900	LIZA	oleo de soja pet

Fonte: Elaborada pelo autor (2025).

4.2.1 Limpeza e Normalização Lexical

Após o carregamento dos dados, a primeira etapa de transformação efetiva foi a limpeza e normalização lexical. Este processo é fundamental para reduzir a variabilidade superficial das descrições textuais, que, embora semanticamente equivalentes, podem apresentar-se de formas distintas. Tais variações, se não tratadas, são interpretadas por algoritmos como diferenças significativas, comprometendo a eficácia de qualquer técnica de correspondência de *strings*. Em essência, esta etapa pode ser compreendida como um processo de redução da dimensionalidade no espaço de características textuais, onde múltiplas representações de um mesmo conceito são mapeadas para uma única forma canônica. Essa padronização simplifica drasticamente o problema para os algoritmos de *matching*, tornando-os mais precisos e eficientes (CHAI, 2022).

O procedimento foi executado em três operações sequenciais:

- Conversão para Minúsculas (*Lowercasing*): Todos os caracteres alfabéticos nos nomes dos produtos foram convertidos para suas formas minúsculas. Esta é uma normalização essencial para garantir a insensibilidade a maiúsculas e minúsculas (*case-insensitivity*). Sem essa etapa, um algoritmo trataria "Arroz" e "arroz" como dois *tokens* distintos, introduzindo uma fonte de erro desnecessária na comparação de produtos.
- Remoção de Diacríticos: Acentuações e outros sinais diacríticos (´ ` ^ ~ ¨ ç) foram removidos de todas as palavras. No contexto da língua portuguesa, esta é uma etapa crítica de padronização. A transformação de "Açúcar" para "Acucar", por exemplo, assegura que as comparações textuais não sejam afetadas por variações de acentuação, comuns em dados de fontes diversas e podem ser tratadas de maneira inconsistente por diferentes algoritmos.
- Remoção de Pontuação e Caracteres Especiais: Caracteres não alfanuméricos, como pontos, vírgulas, parênteses, hifens e outros símbolos (. , ( ) -), foram sistematicamente

eliminados das descrições. A justificativa para esta remoção reside no fato de que tais caracteres raramente agregam valor semântico para a identificação do produto e, frequentemente, atuam como ruído, interferindo nos processos de "tokenização" e no cálculo das métricas de similaridade de *strings*. Para a implementação desta remoção baseada em padrões, foram utilizadas expressões regulares (Regex).

4.2.2 Filtragem Semântica para Redução de Complexidade Computacional

A aplicação de algoritmos de correspondência textual em um grande conjunto de dados impõe um desafio computacional significativo. A comparação par a par de todos os produtos do *dataset*, resulta em uma complexidade de tempo quadrática, da ordem de $O(n^2)$, onde n é o número total de produtos. Para um *dataset* com dezenas ou centenas de milhares de itens, cenário comum em catálogos de supermercados, essa complexidade torna-se proibitiva, exigindo um tempo de processamento e recursos de memória que inviabilizariam a experimentação ágil e a análise iterativa.

Para mitigar este desafio, foi adotada uma estratégia de filtragem semântica, cujo objetivo foi reduzir o escopo do problema a um subconjunto de dados relevante e computacionalmente tratável. Esta decisão metodológica representa um balanço deliberado entre a abrangência da análise e a viabilidade da execução do projeto. A estratégia consistiu em focar a análise em produtos que compõem a cesta básica brasileira, um domínio de alto interesse e com representatividade suficiente para uma análise comparativa robusta das técnicas de *matching*.

O processo de filtragem foi implementado por meio da criação de uma lista de palavras-chave (ou "nuvem de palavras") representativas de itens da cesta básica, incluindo termos como "arroz", "feijao", "oleo", "cafe", "acucar", "farinha", "leite", "manteiga", entre outros. O conjunto de dados principal foi, então, submetido a um filtro que reteve somente os registros cujas descrições continham ao menos uma dessas palavras-chave. Esta abordagem pragmática permitiu uma redução drástica no valor de n, tornando a complexidade quadrática gerenciável e possibilitando a condução de experimentos mais intensivos e rápidos com os diferentes algoritmos de correspondência.

4.2.3 Extração e Padronização de Unidades de Medida

Uma das principais fontes de heterogeneidade nas descrições de produtos é a representação de suas quantidades e unidades de medida. Informações como "1kg", "1 kg", "1000g" e "1.0 KG", embora semanticamente idênticas, são textualmente distintas. Para um sistema computacional poder comparar produtos inteligentemente, é imprescindível que ele consiga compreender e normalizar essa informação quantitativa.

O processo foi dividido em duas fases interdependentes, ambas implementadas

com o auxílio de expressões regulares (Regex) em Python, devido à sua flexibilidade para identificar padrões em texto:

1. **Extração**: Primeiramente, foram desenvolvidos padrões de Regex para identificar e extrair valores numéricos e suas unidades de medida associadas diretamente da *string* de descrição do produto. Os padrões foram projetados para capturar uma ampla variedade de formatos, incluindo números inteiros e decimais, diferentes grafias de unidades ("kg", "g", "l", "ml") e a presença ou ausência de espaços entre o número e a unidade. O resultado desta fase foi a extração de pares de valor e unidade para cada produto onde essa informação estava presente.

2. **Normalização e Padronização**: Uma vez extraídos, os pares de valor e unidade foram submetidos a um processo de normalização. Foi definida uma base de unidades padrão para cada dimensão de medida: massa foi padronizada para gramas (g), volume para mililitros (ml) e comprimento para centímetros (cm). Uma lógica de conversão foi implementada para transformar todas as quantidades extraídas para esta base comum. Por exemplo, "1kg" foi convertido para 1000 (g), "1,5L" para 1500 (ml), e assim por diante. Os valores numéricos normalizados e as unidades padronizadas foram então armazenados em novas colunas dedicadas, transformando a informação antes implícita no texto em características explícitas e quantitativas.

4.2.4 Identificação Híbrida de Marcas

A marca de um produto é uma informação de alta relevância, mas que pode atuar como uma variável de confusão em uma análise de correspondência textual focada na descrição intrínseca do item. Se a marca for mantida no nome do produto, algoritmos de similaridade podem atribuir pontuações altas a pares de produtos simplesmente por compartilharem a mesma marca, sem avaliar a semelhança de suas descrições principais (e.g., "arroz *codil tipo 1*" vs. "feijão *codil tipo 1*"). Além disso, o contrário também pode acontecer, algoritmos de similaridade podem atribuir pontuações baixas a pares de um mesmo produto, simplesmente por compartilharem marcas distintas (e.g., "arroz *codil* tipo 1" vs. "arroz *prato fino* tipo 1". Para garantir que a análise comparativa avaliasse a capacidade dos modelos de compreender a semântica do produto em si, foi necessário primeiro identificar e depois remover a marca da descrição textual.

Dada a complexidade de identificar marcas em um texto livre e não padronizado, foi implementada uma abordagem híbrida que combina eficiência e alta capacidade de cobertura:

1. **Abordagem Baseada em Lista**: Inicialmente, foi compilada uma lista extensa de marcas comuns encontradas em supermercados brasileiros. O sistema realizou

uma busca por essas marcas conhecidas no nome de cada produto. Este método é computacionalmente eficiente e eficaz para a grande maioria dos produtos, cujas marcas são de alta frequência e bem estabelecidas no mercado.

2. **Abordagem Baseada em Inteligência Artificial Generativa**: Para os produtos cujas marcas não foram identificadas na primeira etapa — casos de marcas menos conhecidas, novas no mercado ou com grafias ambíguas —, recorreu-se a uma segunda solução. Foi utilizada a API da OpenAI[5], que fornece acesso a modelos de linguagem de grande escala (LLMs). Para cada descrição de produto remanescente, uma requisição (*prompt*) foi enviada ao modelo, instruindo-o a identificar e extrair o nome da marca contido no texto. O uso de IA generativa para esta tarefa de limpeza de dados demonstra a aplicação de tecnologia para resolver um problema complexo de extração de informação que seria difícil de solucionar somente com regras predefinidas.

Após a identificação da marca por um dos dois métodos, a etapa final e fundamental foi a sua remoção da *string* do nome do produto. O resultado foi uma descrição limpa, contendo somente os atributos essenciais do item (e.g., de "arroz agulhinha tipo 1 *camil* 1kg" para "arroz agulhinha tipo 1 1kg"). A marca identificada foi armazenada em uma coluna separada, preservando a informação, mas isolando-a do campo que seria utilizado para o cálculo da similaridade textual.

4.2.5 Persistência do Conjunto de Dados Tratado

A etapa final do *pipeline* de pré-processamento foi a persistência do conjunto de dados final, agora limpo, estruturado e enriquecido. A conclusão bem-sucedida das fases anteriores resultou em um DataFrame do Pandas contendo não somente as descrições de produtos normalizadas, mas também novas colunas com informações estruturadas, como quantidades padronizadas, unidades de medida e marcas identificadas. Neste ponto, o dado deixa de ser um objeto transiente, em memória, para se tornar um ativo de dados persistente, estável e consultável.

Para o armazenamento, foi escolhido o sistema de gerenciamento de banco de dados relacional PostgreSQL. A decisão de utilizar um banco de dados em detrimento de um arquivo simples, como um CSV, foi estratégica e baseada em melhores práticas de engenharia de dados. As principais justificativas para essa escolha incluem:

- **Integridade dos Dados**: Um SGBD relacional como o PostgreSQL oferece mecanismos robustos para garantir a integridade dos dados, como a definição de tipos

[5] API platform | OpenAI - https://openai.com/pt-BR/api/

de dados para cada coluna, restrições e transações, prevenindo a corrupção e a inconsistência dos dados.

- **Eficiência na Consulta**: A persistência em um banco de dados permite a execução de consultas complexas e eficientes via linguagem SQL, facilitando a seleção de subconjuntos de dados específicos.
- **Escalabilidade e Concorrência**: Bancos de dados são projetados para gerenciar grandes volumes de dados e suportar acessos concorrentes de forma segura, o que torna a solução mais escalável e robusta para futuras expansões do projeto.
- **Separação de Responsabilidades**: A persistência dos dados tratados em uma base de dados demarca uma clara separação entre a etapa de preparação de dados e a etapa de análise e modelagem. Isso cria um fluxo de trabalho mais organizado e modular, onde o *dataset* limpo se torna a principal fonte dos dados, que pode ser consumida por diferentes processos analíticos de forma consistente e reprodutível.

Ao persistir os dados em um banco de dados PostgreSQL, o projeto evolui de um simples *script* de processamento para um *pipeline* de dados mais maduro e profissional, estabelecendo uma base sólida e confiável para a subsequente análise comparativa das técnicas de correspondência textual.

4.3 Aplicação das técnicas de Correspondência Textual

Concluída a etapa de pré-processamento e estruturação dos dados, o desenvolvimento avança para a fase de aplicação das técnicas de correspondência textual. Esta seção detalha a aplicação prática do *pipeline* desenvolvido para identificar produtos idênticos ou similares entre diferentes supermercados, utilizando o *dataset* pré-processado e padronizado.

O desafio central nesta etapa é o problema de *Fuzzy Matching* (correspondência aproximada), exacerbado por inconsistências que persistem mesmo após a limpeza e tratamento dos dados, como o uso de sinonímia, abreviações e inversão de termos (JARO, 1989; WINKLER, 1990).

O pipeline de análise utiliza seis técnicas de similaridade que operam em diferentes níveis de abstração textual, permitindo uma análise multifacetada. O objetivo é analisar os resultados de cada técnica, e observar possíveis combinações entre as técnicas para maximizar os resultados (abordagem híbrida). As técnicas são agrupadas em três categorias:

1. **Abordagens Clássicas**: Métricas baseadas em distância de edição e sobreposição de conjuntos, eficazes na captura de erros tipográficos e inversões (Levenshtein, Jaccard, Jaro-Winkler).

2. **Abordagens Vetoriais**: Técnicas do Modelo de Espaço Vetorial que comparam o conteúdo lexical (Bag-of-Words, TF-IDF com Similaridade de Cossenos).

3. **Abordagens Semânticas**: Métodos modernos baseados em *transformers* que capturam o significado (SBERT), resolvendo o problema da sinonímia.

4.3.1 *Pipeline* de Análise e Otimização por Categoria

O *pipeline* para a aplicação das métricas de similaridade segue um fluxo de execução controlado, projetado para garantir tanto a precisão da análise quanto a viabilidade computacional.

O fluxo de processamento é o seguinte:

1. **Leitura do *Dataset***: O processo inicia-se com a leitura dos dados pré-processados e estruturados, carregados a partir do banco de dados PostgreSQL para um DataFrame Pandas.

2. **Categorização de Produtos**: Antes de qualquer comparação, os produtos são agrupados por categorias (ex: "arroz", "feijão", "leite"). Esta etapa, que utiliza palavras-chave predefinidas para identificar o tipo de produto, é uma otimização metodológica crucial. A alternativa, comparar todos os produtos entre si (N itens), resultaria em uma complexidade de tempo quadrática ($O(N^2)$). Ao segmentar o problema, o *pipeline* executa comparações apenas dentro de cada categoria (contendo aproximadamente M itens cada), o que torna o processamento computacionalmente tratável. Assim, as comparações passam a ocorrer apenas dentro de cada categoria, reduzindo o custo para $O(M^2)$ por categoria. Considerando $K = N/M$ categorias, a complexidade total torna-se $O(K \cdot M^2) = O(\frac{N}{M} \cdot M^2) = O(NM)$, o que é significativamente menor que $O(N^2)$ quando $M \ll N$.

3. **Inicialização do Modelo Semântico**: Uma única instância do modelo Sentence-Transformer (SBERT) é carregada em memória. Foi selecionado o modelo MiniLM (WANG et al., 2020), uma versão leve e otimizada do BERT, que oferece um balanço ideal entre *performance* semântica e eficiência computacional.

4. **Cálculo de Similaridade em Lote**: O *pipeline* itera sobre cada categoria de produto. Dentro de cada grupo, são geradas todas as combinações possíveis de pares únicos de produtos. Para cada par, o *script* calcula e armazena os *scores* de similaridade gerados pelas seis métricas implementadas (Jaccard, Levenshtein, Jaro-Winkler, BoW-Cosseno, TF-IDF-Cosseno e SBERT-Cosseno).

4.3.2 Técnicas de Similaridade Aplicadas

Para a execução da etapa 4 do *pipeline*, foi necessário implementar as seis métricas de similaridade. As subseções seguintes detalham o funcionamento e a aplicação de cada uma dessas técnicas.

4.3.2.1 Distância de Levenshtein

A Distância de Levenshtein, introduzida por Vladimir Levenshtein em seu trabalho sobre correção de erros em códigos binários. A métrica é definida como o número mínimo de operações de edição de caractere único (inserção, deleção ou substituição) necessárias para transformar uma *string* a em uma *string* b. Dessa forma, uma de suas principais características é capturar similaridades baseadas em erros tipográficos (ex: “nescau” vs “nescal”).

A distância é calculada eficientemente através de programação dinâmica, utilizando uma matriz D de dimensões $(|a|+1) \times (|b|+1)$. Cada célula $D(i,j)$ armazena a distância mínima entre os i primeiros caracteres de a e os j primeiros caracteres de b. A matriz é preenchida usando a seguinte relação de recorrência:

$$D(i,j) = \min \begin{cases} D(i-1,j)+1 & \text{(Custo de Deleção)} \\ D(i,j-1)+1 & \text{(Custo de Inserção)} \\ D(i-1,j-1)+\text{custo} & \text{(Custo de Substituição ou Match)} \end{cases}$$

Onde o custo é 0 se $a[i] = b[j]$ (*match*) e 1 caso contrário. O valor final, $D(|a|,|b|)$, representa a distância total.

4.3.2.2 Índice de Jaccard

Para lidar com a inversão de termos (ex: “acucar cristal organico” vs “acucar organico cristal”), que a Distância de Levenshtein penaliza severamente, foi aplicada uma métrica baseada em conjuntos, o Índice de Jaccard (JACCARD, 1901; JACCARD, 1908), $J(A,B)$, mede a similaridade entre dois conjuntos finitos A e B. Ele é definido como a razão entre o tamanho da interseção e o tamanho da união dos conjuntos:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Para a aplicação no *pipeline*, as *strings* de nomes de produtos (já pré-processadas) são primeiro convertidas em conjuntos de *tokens* (palavras). Por exemplo, “acucar cristal organico” torna-se o conjunto {"acucar", "cristal", "organico"}.

Demonstração do cálculo:

- $A =$ “acucar cristal organico” $\rightarrow SetA =$ {“acucar”, “cristal”, “organico”}
- $B =$ “acucar organico cristal” $\rightarrow SetB =$ {“acucar”, “organico”, “cristal”}
- $A \cap B =$ { “acucar”, “organico”, “cristal”} (Tamanho = 3)
- $A \cup B =$ { “acucar”, “organico”, “cristal”} (Tamanho = 3)
- $J(A, B) = \frac{3}{3} = 1.0$

4.3.2.3 Similaridade de Jaro-Winkler

A terceira técnica clássica aplicada foi a Similaridade de Jaro-Winkler (JARO, 1989; WINKLER, 1990). Esta métrica foi desenvolvida especificamente para o *record linkage* no Censo dos EUA e é robusta contra transposições de caracteres, além de valorizar prefixos comuns. A métrica é uma evolução da Similaridade de Jaro (S_j), que é calculada como:

$$S_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right)$$

Onde:

- m é o número de caracteres correspondentes (idênticos e dentro de uma janela de distância máxima).
- t é a metade do número de transposições (caracteres correspondentes, mas fora de ordem).
- $|s_1|$ e $|s_2|$ são os comprimentos das strings.

Winkler (1990) aperfeiçoou esta métrica adicionando um bônus de prefixo, baseado na observação empírica de que erros são menos comuns no início das strings. A Similaridade de Jaro-Winkler (S_w) é calculada como:

$$S_w = S_j + (L \cdot P \cdot (1 - S_j))$$

Onde:

- S_j é a pontuação de Jaro.
- L é o comprimento do prefixo comum, com um máximo de 4 caracteres.
- P é um fator de escala constante, tipicamente 0.1.

Esta métrica foi aplicada por ser ideal para nomes de produtos, onde os identificadores principais (tipo de produto) frequentemente aparecem no início do texto.

4.3.2.4 Modelagem Vetorial (BoW e TF-IDF)

As abordagens clássicas operam no nível de caracteres ou conjuntos de palavras. Para capturar a importância relativa das palavras em uma descrição, o pipeline implementou também o Modelo de Espaço Vetorial (VSM), uma abordagem seminal da recuperação de informação (DIERK, 1972; SALTON; WONG; YANG, 1975).

Neste modelo, os nomes dos produtos são convertidos em vetores numéricos, onde cada dimensão corresponde a um termo (palavra) do vocabulário daquela categoria de produtos. Duas variantes de ponderação foram aplicadas:

- **Bag-of-Words (BoW)**: É o método de vetorização mais direto, onde o valor de cada dimensão do vetor é simplesmente a contagem (frequência) daquela palavra no nome do produto. Este modelo desconsidera a ordem, tratando o texto como um "saco de palavras".
- **TF-IDF (Term Frequency-Inverse Document Frequency)**: O BoW trata palavras comuns (ex: "leite") e palavras distintivas (ex: "lactose") com o mesmo peso. O TF-IDF corrige isso atribuindo um peso maior a termos que são simultaneamente frequentes em um documento (alto TF) e raros em todos os outros documentos da categoria (alto IDF). O peso $w_{t,d}$ do termo t no documento d é:

$$w_{t,d} = \mathrm{TF}_{t,d} \times \mathrm{IDF}_t$$

Onde $\mathrm{IDF}_t = \log\left(\frac{|M|}{|\{d' \in M | t \in d'\}|}\right)$, sendo $|M|$ o número total de produtos na categoria.

Ao aplicar ambas as técnicas, o *pipeline* gera duas representações vetoriais distintas para cada produto: uma baseada na contagem (BoW) e outra baseada na relevância (TF-IDF).

4.3.2.5 Cálculo Vetorial: Similaridade de Cossenos

Após a vetorização dos nomes dos produtos (via BoW e TF-IDF), foi necessário aplicar uma métrica para comparar os vetores resultantes. A Similaridade de Cossenos mede o cosseno do ângulo θ entre dois vetores x e y em um espaço vetorial. Ela é ideal para VSM, pois foca exclusivamente na orientação (direção) dos vetores, ignorando suas magnitudes (KRANTZ; JONKER, 2025).

A fórmula é derivada do produto escalar:

$$\text{Similaridade}(x, y) = \cos(\theta) = \frac{x \cdot y}{||x|| \cdot ||y||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}}$$

O *pipeline* calcula a Similaridade de Cossenos para ambos os tipos de vetores, resultando em duas pontuações distintas: BoW-Cosseno (similaridade genérica) e TF-IDF-Cosseno (similaridade distintiva).

4.3.2.6 Abordagem Semântica (SBERT/MiniLM)

Todas as técnicas anteriores são lexicais: falham se dois produtos usam sinônimos (ex: "leite uht" vs "leite longa vida"). Para capturar o significado (semântica), foi aplicado no *pipeline* uma técnica moderna baseada em *Transformers*. O modelo BERT (DEVLIN et al., 2019) é um *cross-encoder*: para comparar um Par (x, y), ele exige que ambos sejam processados simultaneamente. Em uma categoria com n produtos, isso exigiria $O(n^2)$ inferências do modelo, um custo computacional proibitivo para este TCC. Para solucionar isso, foi aplicado o Sentence-BERT (SBERT) (REIMERS; GUREVYCH, 2019). O SBERT utiliza uma arquitetura Bi-Encoder (rede siamesa) que gera *embeddings* de sentenças (vetores de significado) de forma independente. Tendo a seguinte aplicação na *pipeline*:

1. **Geração de Embeddings**: Na etapa 3 do *pipeline*, o SBERT processa cada um dos M produtos da categoria uma única vez, gerando M vetores (*embeddings*) que capturam seu significado.
2. **Comparação por Similaridade de Cossenos**: Na etapa 4, as comparações $O(M^2)$ são reduzidas a cálculos de Similaridade de Cossenos entre os vetores pré-calculados, o que é computacionalmente trivial.

Para executar esta etapa eficientemente, o *pipeline* utiliza o *all-MiniLM-L6-v2*. Este modelo, desenvolvido pela Microsoft (WANG et al., 2020), é um *transformer* leve obtido por destilação de conhecimento (*knowledge distillation*). Ele "destila" o conhecimento de um modelo professor (grande) para um modelo estudante (pequeno), focando no módulo de auto-atenção da última camada. O MiniLM mantém alta precisão semântica com uma fração do custo computacional, tornando a análise semântica viável no escopo deste trabalho.

4.3.3 Quadro Comparativo das Técnicas Aplicadas

A Tabela 5 apresenta um resumo comparativo das seis técnicas de correspondência implementadas no *pipeline* de análise. Esta tabela sintetiza o principal foco de cada técnica, mostrando as diferenças e semelhanças no momento de comparar os produtos. Dessa forma, fica evidente que a aplicação de uma abordagem híbrida entre mais de uma técnica pode apresentar resultados interessantes.

Tabela 5 – Quadro Comparativo das Técnicas de Similaridade Aplicadas

Técnica	**Base de Comparação**	**Foco Principal**
Levenshtein	Caractere (Distância de Edição)	Captura de erros de digitação e pequenas variações lexicais.
Jaccard	Conjunto (*Token* de Palavra)	Captura de inversões de ordem de palavras, sendo insensível à posição.
Jaro-Winkler	Caractere (Transposições e Prefixo)	Otimizada para *record linkage*, captura transposições e valoriza correspondências no prefixo.
BoW + Cosseno	Vetorial (Contagem)	Mede a similaridade *genérica* de conteúdo, ponderando todos os termos de forma igual (normalizado pelo cosseno).
TF-IDF + Cosseno	Vetorial (Relevância)	Mede a similaridade *distintiva*, dando maior peso a termos-chave (raros) e penalizando termos comuns.
SBERT (MiniLM)	Vetorial (*Embeddings*)	Captura de similaridade semântica (significado), resolvendo o problema de sinonímia.

Fonte: Elaborada pelo autor (2025).

5 Resultados e Análise

Esta seção apresenta a análise quantitativa e qualitativa do desempenho das seis técnicas de correspondência textual aplicadas ao *dataset* de produtos de supermercado coletado. A avaliação baseia-se na comparação das métricas de Precisão, Revocação (*Recall*) e F1-Score, utilizando como referência (*ground truth*) a base de consenso gerada a partir da concordância entre múltiplos algoritmos.

5.1 Resultados da coleta de dados (*Web scraping*)

A etapa de coleta de dados foi realizada em quatro plataformas de *e-commerce* do setor supermercadista, selecionadas por sua representatividade no mercado online e pela disponibilidade pública de catálogos de produtos, são elas:

- Supermercados ABC | Atacado e Varejo[1]
- Supermercados Rena[2]
- Supermercados SuperSo[3]
- Supermercados Carrefour[4]

O processo de raspagem coletou, no total bruto, 35.618 registros de produtos distribuídos entre as quatro fontes supra-citadas. Porém, para viabilizar a etapa experimental de comparação entre técnicas de correspondência textual, adotou-se um recorte amostral focado em produtos pertencentes a uma Cesta Básica de Alimentos[5]. Esse filtro resultou na seleção de 6.865 (seis mil, oitocentos e setenta e cinco) produtos, os quais compuseram a base utilizada nas fases de pré-processamento e limpeza e, em seguida, na comparação entre as seis técnicas aplicadas, os detalhes do filtro aplicado está na seção 4.2.2. A seleção por cesta básica justificou-se por:

- Reduzir a variabilidade conceitual entre categorias;
- Concentrar o experimento em itens de alta rotatividade e importância prática para consumidores e pesquisadores de varejo;

[1] https://www.superabc.com.br/
[2] https://www.renaemcasa.com.br/p
[3] https://www.superso.com.br/
[4] https://mercado.carrefour.com.br/
[5] Cesta Básica de Alimentos é o conjunto de alimentos que busca garantir o direito humano à alimentação adequada e saudável, à saúde e ao bem-estar da população. Conforme https://www.gov.br/mds/pt-br/acoes-e-programas/promocao-da-alimentacao-adequada-e-saudavel/cesta-basica-de-alimentos

- Reduzir a complexidade computacional, viabilizando o desenvolvimento deste trabalho.

A tabela Tabela 6 apresenta o perfil do *dataset* conforme os números apresentados e os totais calculados a partir das contagens por categoria.

Tabela 6 – Perfil do *Dataset* - Resumo dos dados coletados

Métrica	**Valor**
Total de produtos coletados	35.618
Total de produtos selecionados	6.865
Total de categorias	21
Média de produtos selecionados por categoria	327

Fonte: Elaborada pelo autor (2025).

5.2 Agrupamento dos Produtos por Técnica Analisada

A partir do *dataset*, foi feita a categorização dos produtos e aplicado as técnicas de correspondência textual, como descrito na seção de desenvolvimento, obtendo a similaridade para cada par de produtos, ao final, foram comparados 1.892.423 pares de produtos. Então implementou-se um algoritmo de agrupamento baseado em teoria de grafos que transforma pares similares em um grafo não-direcionado e extrai componentes conectadas como *clusters* de produtos. O propósito é transformar pares de alta similaridade (*match pairwise*) em grupos coerentes (*entity clusters*). Para isso, foi seguido o fluxo de processamento:

1. **Filtragem de pares**: selecionar apenas pares cujo limiar de similaridade seja maior ou igual a 0,90.
2. **Construção do grafo**: criar um grafo não-direcionado, utilizando a biblioteca *python NetworkX*, onde cada nó representa um produto e cada aresta representa uma relação de similaridade entre dois nós.
3. **Extração de componentes**: identificar componentes conectadas, cada componente é tratada como um cluster de produtos.
4. **Formatação de saída**: converter os clusters encontrados para JSON.

A tabela Tabela 7 mostra a quantidade de *clusters* formados a partir do fluxo de processamento de agrupamento dos produtos. Cada *cluster* consiste em um grupo de produtos de alta similaridade que consiste uma mesma entidade (grupo coerente).

Tabela 7 – Quantidade de *Clusters* Formados por Técnica de *Matching*

Técnica	**Quantidade de *Clusters***
TF-IDF	1.255
Jaccard	1.225
Levenshtein	1.276
Bag of Words	1.311
SBERT	1.213
Jaro-Winkler	626

Fonte: Elaborada pelo autor (2025).

Analisando a quantidade de clusters gerados para cada técnica, é interessante observar que *Jaro-Winkler* obteve um número bem abaixo de grupos quando comparado com o restante das técnicas.

5.3 Limitações na base *ground truth*

Uma limitação deste trabalho está na construção da base de verdade (*Ground Truth*). Devido à inviabilidade de rotular manualmente milhares de pares de produtos, utilizou-se uma abordagem de Padrão Prata (*Silver Standard*), gerada pelo consenso entre as próprias técnicas avaliadas. Esta metodologia, embora comum em *Big Data*, introduz vieses epistemológicos que devem ser considerados na interpretação dos dados (VOGEL et al., 2014).

A base de consenso, gerada neste trabalho para atuar como *ground truth*, foi criada por meio da implementação de um algoritmo que comparou os grupos de produtos resultantes de cada técnica aplicada, e realizou a intercessão para identificar produtos verdadeiramente similares. Para cada produto do *dataset*, o *script* mapeia em quais grupos ele aparece nas diferentes técnicas de *matching*, consolidando todos os produtos que compartilham os mesmos grupos. Em seguida, aplica um filtro de frequência, mantendo apenas produtos que aparecem 4 ou mais vezes nos grupos consolidados, o que indica que a maioria das técnicas concorda sobre sua similaridade. Os grupos resultantes são dedupicados e salvos em um novo *dataset*, gerando uma lista de *clusters* de produtos com alta confiabilidade de similaridade, validada pelo consenso entre as diferentes técnicas de *matching*. Ao final deste processamento, foi obtido 1.314 *clusters*.

5.4 Panorama Geral dos Resultados

A Tabela 8 resume as métricas obtidas para cada técnica. As técnicas foram ordenadas de forma decrescente pelo F1-Score, que representa a média harmônica entre Precisão e Revocação.

Tabela 8 – Desempenho Comparativo das Técnicas de *Matching*

Técnica	TP (Verdadeiros)	FP (Falsos)	FN (Perdidos)	Precisão (P)	Revocação (R)	F1-Score
TF-IDF	7.338	26	64	0,9965	0,9913	**0,9939**
Jaccard	7.014	**2**	144	**0,9997**	0,9799	0,9897
Levenshtein	7.108	804	172	0,8984	0,9764	0,9358
Bag of Words	7.600	2.332	**0**	0,7652	**1,0000**	0,8670
SBERT	7.594	40.152	6	0,1590	0,9992	0,2744
Jaro-Winkler	7.568	227.084	30	0,0322	0,9960	0,0625

Fonte: Elaborada pelo autor (2025).

Analisando-se os resultados da Tabela 8, observa-se um *trade-off* claro entre as técnicas:

- **Bag of Words** atingiu a Revocação perfeita (1,0), recuperando 100% dos itens do *Ground Truth*, mas gerou ruído significativo (2.332 Falsos Positivos).
- **TF-IDF** emergiu como a técnica mais equilibrada (F1 = 0,9939), quase eliminando os falsos positivos sem sacrificar significativamente a revocação.
- **Jaccard** demonstrou uma alta precisão (99,97%), errando somente duas associações em todo o experimento.

Para facilitar a visualização da disparidade de eficiência entre os modelos, apresenta-se a Figura 4 com um comparativo de F1-Score.

5.5 Análise das Técnicas *Jaccard*, TF-IDF e *Bag-of-Words*

As técnicas que operam sobre a presença e frequência de *tokens* (palavras) apresentaram os melhores resultados globais.

O Índice de *Jaccard* obteve o melhor desempenho geral (F1 = 0,9893), gerando apenas 144 Falsos Positivos. O TF-IDF seguiu muito próximo (F1 = 0,9812). A alta eficácia desses métodos sugere que os nomes de produtos em supermercados são curtos, estruturados e possuem termos distintivos claros (ex: tipos de produto, sabores, classificações). O *Jaccard*, ao medir a interseção de conjuntos, e o TF-IDF, ao penalizar termos muito comuns (como "de", "com", "sem", "tipo"), conseguem isolar as palavras-chave que definem a identidade do produto.

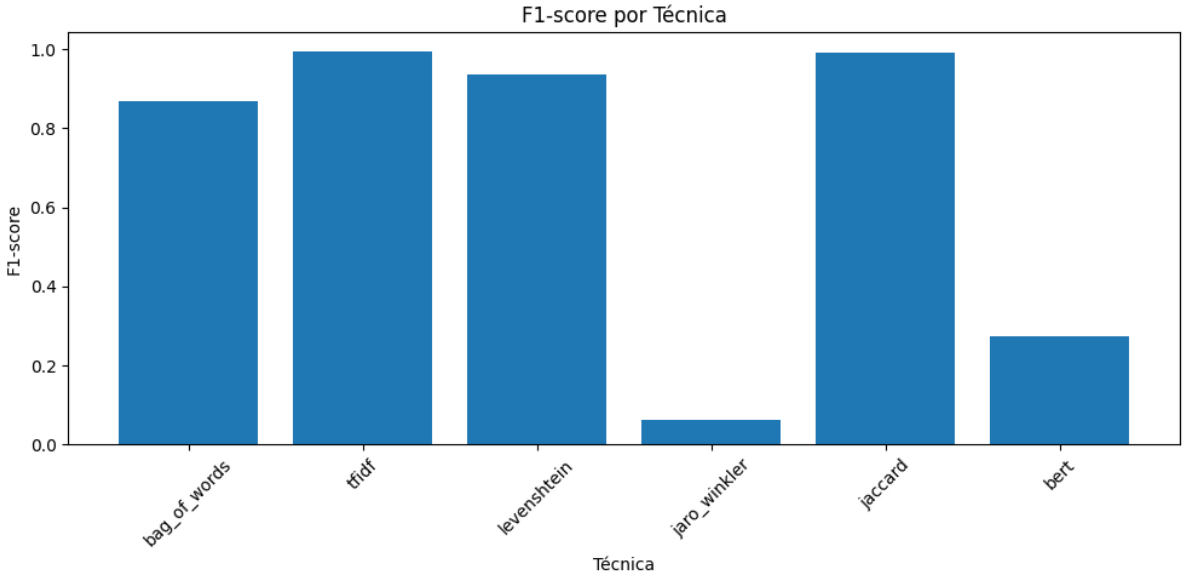

Figura 4 – Gráfico comparativo de F1-Score por técnica.

Fonte: Elaborada pelo autor.

Já a técnica *Bag-of-Words*, embora pertença à mesma família, teve desempenho inferior (F1 = 0,86). A ausência de ponderação (como no TF-IDF) ou normalização por união (como no *Jaccard*) tornou o BoW sensível à repetição de palavras comuns, aumentando os Falsos Positivos.

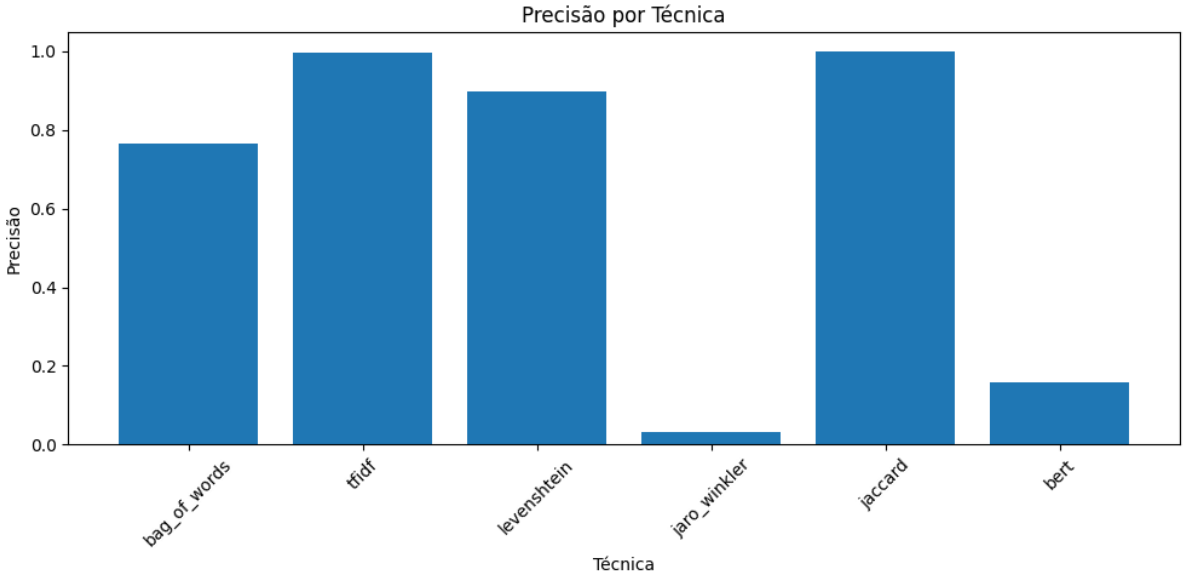

Figura 5 – Valores de Precisão por técnica.

Fonte: Elaborada pelo autor.

A Figura 5 ilustra a Precisão de cada técnica, reforçando como o *Jaccard* e o TF-IDF maximizam a correção das correspondências textuais.

5.6 Análise das Técnicas de Distância de Levenshtein e Jaro-Winkler

Este grupo de técnicas avalia a similaridade baseada na alteração de caracteres, apresentando resultados mistos.

A distância de *Levenshtein* mostrou-se robusta (F1 = 0,9392), superando o *Bag of Words*. O método foi eficaz para lidar com variações morfológicas e pequenos erros de digitação comuns em cadastros manuais. No entanto, observou-se que o algoritmo falha ao distinguir produtos que diferem em alguns detalhes (como quantidade de unidades, classificações), pois a distância de edição entre essas *strings* é mínima, embora sejam produtos distintos.

Já o algoritmo *Jaro-Winkler* apresentou o pior desempenho do estudo (F1 = 0,0664), com uma taxa massiva de Falsos Positivos. O baixo desempenho deve-se ao viés de prefixo (*prefix bias*) do algoritmo, que aumenta a pontuação para *strings* que começam iguais. Como muitos produtos iniciam com a mesma categoria ou marca (ex: "Biscoito...", "Refrigerante...", "Café..."), o *Jaro-Winkler* classificou erroneamente milhares de produtos distintos como similares apenas por compartilharem o início do nome. A Tabela 9 e Tabela 10 mostram dois *clusters* gerados pela técnica *Jaro-Winkler*, onde é possível observar o *match* em produtos diferentes, por terem o mesmo prefixo. Outros exemplos podem ser encontrados no Apêndice A.2 ilustrando clusters completos de produtos agrupados por terem um mesmo prefixo, mesmo sendo produtos distintos.

Tabela 9 – *Cluster* de produtos do tipo "Refrigerante", agrupados pela técnica *Jaro-Winkler*

Produto
refrigerante guarana zero acucar garrafa
refrigerante zero acucar lata
refrigerante sem acucar 1,5l
refrigerante lemon fresh sem acucar garrafa
refrigerante menos acucar 2,5l
refrigerante black zero acucar garrafa

Fonte: Elaborada pelo autor (2025).

A magnitude do erro do *Jaro-Winkler* pode ser visualizada na Figura 6, que mostra o gráfico de Falsos Positivos de todas as técnicas.

5.7 Análise da Técnica Semântica SBERT

O modelo SBERT, baseado em *Deep Learning*, obteve um desempenho insatisfatório para a tarefa de *matching* exato de produtos (F1 = 0,2956, precisão = 17,34%) utilizando

Tabela 10 – *Cluster* de produtos do tipo “Biscoito”, agrupados pela técnica *Jaro-Winkler*

Produto
biscoito cookie integral banana zero acucar pacote
biscoito cookie integral zero acucar ameixa e coco pacote
biscoito recheado de morango zero acucar e lactose
biscoito recheado de limao zero acucar e lactose
biscoito rosquinha integral coco zero acucar pacote

Fonte: Elaborada pelo autor (2025).

o limiar de 0,90. Diferente das técnicas léxicas, o SBERT agrupa textos baseando-se em similaridade semântica (significado). No contexto de produtos de supermercado, isso gerou *alucinações* de similaridade: o modelo agrupou produtos diferentes (ex: “leite em pó integral” e “leite em pó desnatado”) porque ambos são semanticamente “leite em pó”. Embora o modelo tenha *entendido* o contexto, ele falhou na distinção de atributos específicos (sabor, tipo, peso) que definem as características do produto.

Além disso, vale ressaltar que é provável que uma parte desses pares sejam correspondências semânticas legítimas (ex: sinônimos como “gelatina zero acucar uva” e “gelatina em po uva zero acucar pacote”) que não possuem sobreposição léxica. Como o *Ground Truth* foi gerado por consenso léxico, essas conexões semânticas não foram incluídas na base de Verdadeiros Positivos (TP). Consequentemente, o SBERT foi “punido” por detectar similaridades que os métodos clássicos não conseguiram ver. A tabela Tabela 11

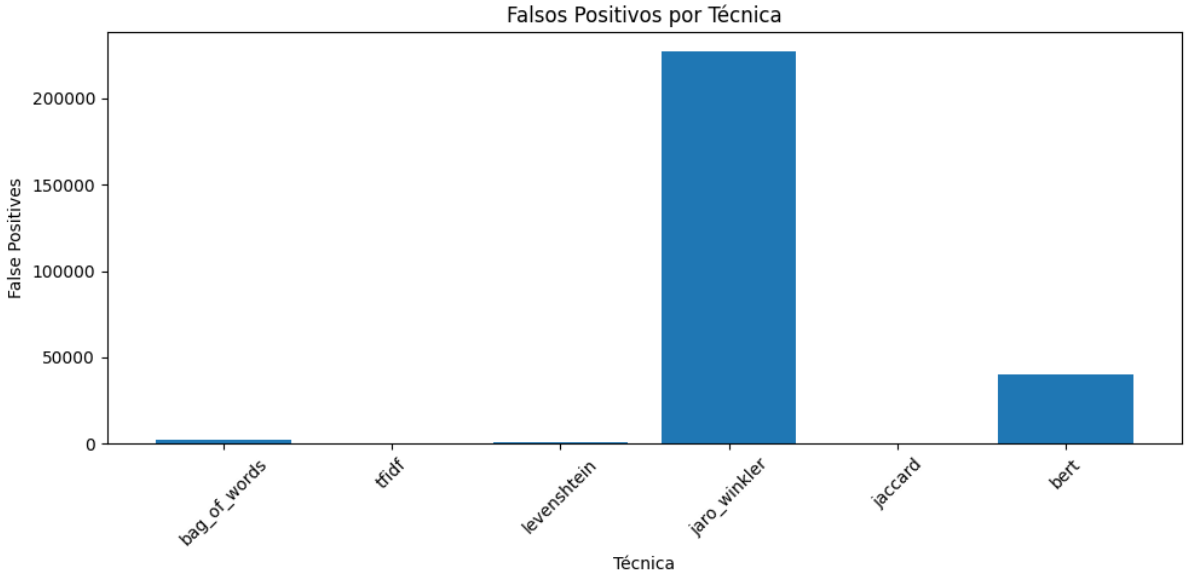

Figura 6 – Quantidade absoluta de Falsos Positivos.

Fonte: Elaborada pelo autor.

mostra pares de produtos agrupados pela técnica SBERT, mas que o *dataset Ground Truth* indicou como produtos distintos. Outros exemplos podem ser encontrados no Apêndice A.1, ilustrando uma amostra de clusters formados pela técnica SBERT em que produtos com semelhanças semânticas foram agrupados, mas que, na prática, são produtos distintos.

Tabela 11 – Pares de Produtos Semânticos Agrupados pela Técnica SBERT

Produto 1	**Produto 2**	**SBERT**	***Ground Truth***
gelatina zero acucar uva	gelatina po uva zero acucar pacote	Sim	Não
creme dental menta refrescante protecao bioativa contra o acido do acucar	creme dental protecao bioativa contra o acido do acucar	Sim	Não
vinagre de cereal sabor arroz garrafa	vinagre de cereal arroz frasco	Sim	Não
racao racas pequenas sabor frango e arroz	racao para caes adultos racas pequenas frango arroz	Sim	Não
biscoito recheio goiabinha	biscoito recheio goiabinha barrinha	Sim	Não

Fonte: Elaborada pelo autor (2025).

Por fim, outro fator crítico é que o BERT *básico* (pré-treinado genericamente) pode não possuir a representação vetorial adequada para o jargão específico do domínio deste TCC. Sem um processo de *Fine-Tuning* no *corpus* específico, o modelo usa pesos genéricos que podem não distinguir sutilizas técnicas importantes.

5.7.1 Ajuste de limiar para a técnica SBERT

Para compreender o baixo desempenho inicial do SBERT, foram testados diferentes limiares de similaridade (variando de 0,70 a 0,99, além do 0,90 originalmente utilizado). A Figura 7 mostra os resultados de *F1-Score* para cada limiar aplicado, sendo ajustado somente na técnica SBERT.

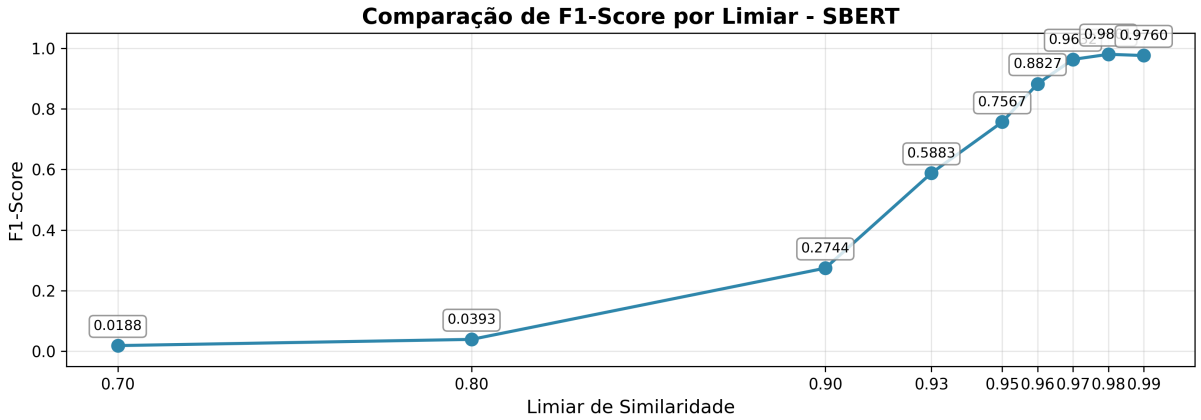

Figura 7 – Curva de evolução do F1-Score do SBERT por limiar de similaridade.

Fonte: Elaborada pelo autor.

Os resultados mostraram que limiares mais baixos, como 0,70 e 0,80, geraram *clusters* excessivamente amplos, agrupando produtos distintos que compartilhavam apenas

alguma relação semântica. Esse comportamento aumentou substancialmente os falsos positivos, resultando em *F1-Scores* baixos: limiar de 0,70 ($F1 = 0,0188$); limiar de 0,80 ($F1 = 0,0393$).

No limiar 0,90 ($F1 = 0,2744$), como mostra a curva da Figura 7, os *clusters* ainda permaneceram maiores que os do *ground truth*, evidenciando que o SBERT, sem ajustes específicos ao domínio, tende a interpretar variações lexicais como equivalências.

Conforme pode ser observado na Figura 7, a situação muda drasticamente nos limiares superiores. Ao aplicar o limiar 0,95 ($F1 = 0,7567$), o resultado já se mostra 175% melhor que o limiar anterior. Contudo, o desempenho máximo foi atingido no limiar 0,98, alcançando um *F1-Score* de 0,9830. Nesse nível, apenas produtos fortemente semelhantes permaneceram agrupados, reduzindo drasticamente os erros semânticos e tornando o modelo competitivo com as técnicas léxicas.

Dessa forma, pode-se concluir que o SBERT, quando aplicado com um mesmo valor de limiar das demais técnicas analisadas (0,90), não é adequado para correspondência precisa de produtos de supermercado. Contudo, com os ajustes realizados para um limiar mais rigoroso (0,98), a técnica se torna altamente consistente. Tal resultado confirma que é possível obter melhorias expressivas no *F1-Score* ao realizar a calibração do limiar de similaridade para o *dataset* específico.

6 Considerações Finais

O presente trabalho teve como propósito investigar e comparar diferentes técnicas de correspondência textual aplicadas ao domínio de produtos de supermercado, um ambiente caracterizado por alta heterogeneidade, ruído textual e ausência de identificadores únicos. A hipótese inicial sustentava que técnicas semânticas baseadas em *transformers* (como o SBERT) apresentariam desempenho superior às técnicas clássicas e vetoriais, por sua capacidade de capturar o significado contextual das descrições dos produtos.

No entanto, os resultados obtidos refutaram parcialmente essa hipótese, demonstrando que, para tarefas de *matching* exato em dados reais e heterogêneos, as técnicas clássicas (*Jaccard* e *Levenshtein*) e vetoriais (TF-IDF com Similaridade de Cossenos) apresentaram desempenho mais eficaz e consistente utilizando uma configuração padrão de limiar de similaridade (0,9), especialmente em cenários em que pequenas variações léxicas definem diferenças cruciais entre os produtos (peso, sabor, marca, tipo, etc).

É fundamental ressaltar, contudo, que a metodologia de construção do *Ground Truth* introduziu um viés analítico importante. Como a base de verdade foi gerada a partir do consenso entre as técnicas avaliadas, e considerando que a maioria destas (cinco das seis) opera sob paradigmas léxicos ou vetoriais, o *Ground Truth* resultante privilegiou naturalmente as correspondências com sobreposição textual explícita. Consequentemente, é provável que o SBERT tenha sido penalizado nas métricas de desempenho ao identificar correlações puramente semânticas que foram rejeitadas pelo "voto da maioria" das técnicas clássicas.

Ainda assim, ao aplicar o SBERT com o mesmo limiar padrão de similaridade (limiar de 0,9) adotado para as demais técnicas, observou-se que a técnica gerou clusters excessivamente amplos, agrupando produtos semanticamente relacionados, porém distintos comercialmente. Essa tendência resultou em numerosos falsos positivos, decorrentes da forma como o modelo interpreta relações semânticas amplas sem considerar detalhes técnicos específicos do domínio.

Para compreender esse comportamento, foram realizados experimentos adicionais variando-se o limiar de similaridade para 0,70, 0,80 e 0,95 (além do 0,90 previamente configurado). Os resultados evidenciaram que limiares mais baixos (0,70 e 0,80) ampliaram ainda mais os clusters, piorando o desempenho devido ao excesso de agrupamentos indevidos. A principal mudança ocorreu com o limiar 0,95, que tornou os agrupamentos mais restritos e semanticamente coerentes. Nesse nível, o SBERT passou a agrupar apenas produtos fortemente similares, reduzindo significativamente os falsos positivos e apresentando melhora substancial no *F1-score*. Esse resultado demonstra que o desempenho do modelo

é altamente sensível ao limiar aplicado, e que limiares mais elevados são essenciais para reduzir a generalização excessiva inerente aos modelos semânticos. Portanto, recomenda-se que tanto para o SBERT quanto para demais técnicas a serem utilizadas, para cada *dataset* específico seja feita uma cuidadosa calibração (*fine-tuning*) do limiar de clusterização (*e.g.*, via projeto fatorial 2^k ou automatizadamente utilizando ferramentas como o iRace[1] ou similar).

Dessa forma, a hipótese inicial é refutada ao se utilizar limiares mais baixos, mas encontra suporte parcial quando aplicado um limiar mais rigoroso, como 0,95. O SBERT revela potencial para aplicações futuras, desde que associado a ajustes criteriosos e otimização de limiar ou a processos de *fine-tuning* específicos para o domínio de supermercado.

Essa constatação abre espaço para abordagens híbridas e estratégias supervisionadas que combinem a riqueza semântica dos modelos *transformers* com a precisão das métricas clássicas e vetoriais utilizadas neste estudo.

6.1 Alcance dos Objetivos

Todos os objetivos propostos na etapa metodológica foram cumpridos com êxito, conforme descrito na estrutura do trabalho:

- Foi realizada uma revisão bibliográfica abrangente sobre *web scraping*, PLN e técnicas de similaridade textual;
- Foi coletado um *dataset* real por meio de *web scraping* em quatro fontes distintas de *e-commerce*, disponibilizado no Kaggle;
- O processo de pré-processamento e normalização dos dados textuais foi realizado, adequando-os ao *pipeline* de PLN;
- Seis técnicas de correspondência textual foram implementadas e aplicadas — três clássicas, duas vetoriais e uma semântica;
- O *ground truth* foi construído manualmente e serviu como base para avaliação do desempenho;
- As técnicas foram analisadas comparativamente pelos indicadores *Precision*, *Recall* e *F1-Score*.

Assim, o objetivo geral foi atingido, permitindo identificar empiricamente quais abordagens são mais adequadas para sistemas de comparação de preços e deduplicação de produtos dos *e-commerces*.

[1] https://mlopez-ibanez.github.io/irace/

6.2 Principais contribuições

Este trabalho gerou *insights* interessantes sobre as técnicas de correspondência textual, e também sobre *web scraping* de *e-commerces* de supermercado, tanto no campo acadêmico quanto no tecnológico e mercadológico, destacando-se:

- Produção de um *dataset* real e especializado, coletado via *web scraping*, que pode ser reutilizado em futuros estudos de PLN e ER, disponibilizado no Kaggle[2];
- Proposição de um *pipeline* completo e replicável, desde a coleta até a avaliação das técnicas, utilizando ferramentas acessíveis e de código aberto, com o código-fonte disponibilizado via GitHub[3];
- Análise crítica e comparativa das técnicas de correspondência, evidenciando seus limites, vantagens e aplicabilidades práticas;
- Produção do *dataset* final, considerando a técnica TF-IDF que obteve o melhor resultado considerando a métrica *F1-score*, que pode ser utilizado em futuros estudos e aplicações para comparação de preços e estudos mercadológicos, disponibilizado no Kaggle[4];
- Identificação de lacunas para futuras pesquisas, especialmente no uso de limiares (de similaridade) otimizados e combinação híbrida de técnicas;
- Demonstração dos desafios reais do PLN aplicado ao varejo alimentar, aproximando soluções acadêmicas das demandas do mercado.

6.3 Trabalhos futuros

Visando a evolução deste trabalho e a mitigação das limitações encontradas, propõem-se as seguintes linhas de pesquisa e desenvolvimento:

- **Otimização de Limiar**: para aumentar a assertividade, deve-se abandonar o limiar padrão, definido inicialmente com 0,9. Sugere-se a implementação de uma otimização de limiar baseada na análise da curva de Precisão-Revocação. Isso permitirá definir limites de decisão personalizados para cada classe.
- **Soluções Híbridas**: Além de otimizar o limiar para cada técnica, existe também, uma grande possibilidade da aplicação de técnicas de forma híbrida apresentar resultados melhores quando comparados a técnicas aplicadas de forma isolada. Dessa

[2] *Dataset* v1 https://www.kaggle.com/datasets/rafaelneto2/produtos-de-supermercados
[3] Código-fonte - https://github.com/rafaelneto2/product-matching
[4] *Dataset* v2 https://www.kaggle.com/datasets/rafaelneto2/produtos-de-supermercados-agrupados

forma, sugere-se a implementação das técnicas de correspondência textual agrupadas, a fim de maximizar a assertividade dos resultados.

- **Adaptação de Domínio (*Domain-Adaptive Pre-training*)**: Como foi analisado nos resultados, a técnica semântica SBERT não obteve os melhores resultados, como se esperava. Sendo assim, sugere-se a realização de melhorias na aplicação do BERT, sendo necessário a aplicação do do fine-tuning supervisionado. Para que o modelo se adeque ao jargão específico deste trabalho.

- **Arquitetura de Microsserviços**: Propõe-se a refatoração do pipeline atual, para uma arquitetura distribuída baseada em microsserviços. Isso permitiria desacoplar e arquitetar uma melhor estrutura para a aplicação. Dessa forma, seria possível escalar horizontalmente recursos específicos conforme a demanda — por exemplo, aumentar o número de instâncias de *crawlers* sem impactar a *performance* do serviço de inferência dos modelos de NLP.

- **Avaliação de Bancos de Dados Vetoriais e NoSQL**: Expandir a análise comparativa para incluir o desempenho de persistência e recuperação de dados em tecnologias modernas. Sugere-se avaliar o uso do MongoDB e do PGVector (extensão do PostgreSQL) integrados aos modelos de geração de *embeddings*. O objetivo seria analisar a eficiência de algoritmos de busca vetorial (*Approximate Nearest Neighbors*) em cenários de *Big Data*, comparando-os com as abordagens em memória utilizadas neste trabalho.

- **Monitoramento de Preços e *Product Matching***: A expansão para comparação de preços exige um módulo de coleta e reconciliação de dados robusto. Dessa forma, poderia ser utilizado o *dataset* gerado por este trabalho para que as comparações pudessem serem feitas.

Referências

Actowiz Solutions. *Web Scraping eCommerce Data – A Comprehensive Guide*. 2024. Disponível em: https://www.actowizsolutions.com/web-scraping-ecommerce-data-guide.php. Acesso em: 10 out. 2024. Citado na página 21.

AHMED, I. et al. The influence of kanban agile methodology on software project management: A survey method. In: *2024 International Conference on Engineering and Emerging Technologies (ICEET)*. IEEE, 2024. p. 1–6. Disponível em: <http://dx.doi.org/10.1109/ICEET65156.2024.10913653>. Citado na página 38.

AKIRA, A. *Setor de alimentos se destaca no crescimento do e-commerce no Brasil - Varejo S.A*. 2025. [Online; accessed 2025-11-12]. Disponível em: <https://cndl.org.br/varejosa/setor-de-alimentos-se-destaca-no-crescimento-do-e-commerce-no-brasil/>. Citado na página 15.

ANDERSON, J. *Speed Up Your Python Program With Concurrency*. 2024. Acesso em: 8 nov. 2025. Disponível em: <https://realpython.com/python-concurrency/>. Citado na página 48.

BEACH, G. *Product content optimization for retailers: Why it should happen sooner than you think*. 2021. Acesso em: 23 maio 2025. Disponível em: <https://www.productsup.com/blog/product-content-optimization-retailers/>. Citado na página 17.

BHUVA, M. *Web Scraping Statistics & Trends You Need To Know In 2025*. 2025. Acesso em: 23 maio 2025. Disponível em: <https://kanhasoft.com/blog/web-scraping-statistics-trends-you-need-to-know-in-2025/>. Citado na página 18.

BOER, Y. D. *Product Matching Software*. 2024. Acesso em: 23 maio 2025. Disponível em: <https://www.lengow.com/solutions/netrivals/product-matching-software/>. Citado na página 17.

BOLTON, D. *Asynchronous programming in Python: A walkthrough*. 2024. Acesso em: 8 nov. 2025. Disponível em: <https://www.dice.com/career-advice/asynchronous-programming-python-walkthrough>. Citado na página 48.

BRIZAN, D. G.; TANSEL, A. U. A. survey of entity resolution and record linkage methodologies. *Communications of the IIMA*, John M. Pfau Library, California State University San Bernardino, v. 6, n. 3, jan. 2015. ISSN 1941-6687. Disponível em: <http://dx.doi.org/10.58729/1941-6687.1324>. Citado na página 26.

CARVALHO, V. D. H. D.; COSTA, A. P. C. S. Exploring text mining and analytics for applications in public security: an in-depth dive into a systematic literature review. *Socioeconomic Analytics*, Even3, v. 1, p. 5–55, jul. 2023. ISSN 2965-4661. Disponível em: <http://dx.doi.org/10.51359/2965-4661.2023.259008>. Citado na página 25.

CASTILLO, D. *Fast and async in Python: Accelerate your requests using asyncio*. 2024. Acesso em: 8 nov. 2025. Disponível em: <https://dylancastillo.co/fast-and-async-in-python-accelerate-your-requests-using-asyncio/>. Citado na página 48.

CHAI, C. P. Comparison of text preprocessing methods. *Natural Language Engineering*, Cambridge University Press (CUP), v. 29, n. 3, p. 509–553, jun. 2022. ISSN 1469-8110. Disponível em: <http://dx.doi.org/10.1017/S1351324922000213>. Citado na página 51.

CHEN, F. Research on real-time e-commerce price comparison system using python web scraping technology. *International Journal of Computer Science and Information Technology*, Warwick Evans Publishing, v. 4, n. 2, p. 127–136, out. 2024. ISSN 3005-9682. Disponível em: <http://dx.doi.org/10.62051/ijcsit.v4n2.18>. Citado na página 18.

CHRISTEN, P. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Berlin, Heidelberg: Springer, 2012. ISBN 9783642311642. Disponível em: <http://dx.doi.org/10.1007/978-3-642-31164-2>. Citado 2 vezes nas páginas 31 e 32.

CHRISTOPHIDES, V.; EFTHYMIOU, V.; STEFANIDIS, K. *Entity Resolution in the Web of Data*. Springer International Publishing, 2015. ISSN 2691-2031. ISBN 9783031794681. Disponível em: <http://dx.doi.org/10.1007/978-3-031-79468-1>. Citado na página 18.

COSTA, V. *E-commerce de Alimentos e Bebidas cresce e se consolida no Brasil - E-Commerce Brasil*. 2025. [Online; accessed 2025-11-12]. Disponível em: <https://www.ecommercebrasil.com.br/artigos/e-commerce-de-alimentos-e-bebidas-cresce-e-se-consolida-no-brasil>. Citado na página 15.

DAHULE, P. The strategic impact of project management and kanban in enhancing data analysis efficiency. *The Eastasouth Journal of Information System and Computer Science*, PT. Sanskara Karya Internasional, v. 1, n. 02, p. 118–125, dez. 2023. ISSN 3026-6041. Disponível em: <http://dx.doi.org/10.58812/esiscs.v1i02.494>. Citado na página 38.

DALVI, A. et al. A comparative analysis of models for dark web data classification. In: _____. *Proceedings of International Joint Conference on Advances in Computational Intelligence*. Springer Nature Singapore, 2024. p. 245–257. ISBN 9789819701803. Disponível em: <http://dx.doi.org/10.1007/978-981-97-0180-3_20>. Citado na página 29.

DEEPANSHI. *Text Preprocessing in NLP with Python Codes*. 2025. Analytics Vidhya. Disponível em: https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/. Acesso em: 10 out. 2024. Citado 2 vezes nas páginas 23 e 24.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Minneapolis, USA: Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: <https://doi.org/10.18653/v1/N19-1423>. Citado 2 vezes nas páginas 30 e 60.

DEWILDE, B. *textacy: API Reference*. [S.l.], 2024. Acesso em: 10 out. 2024. Citado na página 28.

DIERK, S. F. The smart retrieval system: Experiments in automatic document processing — gerard salton, ed. (englewood cliffs, n.j.: Prentice-hall, 1971, 556 pp., 15.00).*IEEE Transactions on Prof* $15, n.\ 1, p.17 - -17, 1972.$*Disponívelem* : <>. Citado na página 59.

DILMEGANI, C. *Roadmap to Web Scraping: Methods & Tools*. 2025. AIMultiple. Acesso em: 23 maio 2025. Disponível em: <https://research.aimultiple.com/web-scraping/>. Citado na página 18.

DILMEGANI, C. *Web scraping challenges*. 2025. Acesso em: 8 nov. 2025. Disponível em: <https://research.aimultiple.com/web-scraping-challenges/>. Citado 3 vezes nas páginas 21, 22 e 47.

DUSEK, V. *Python Playwright: the ultimate guide*. 2024. Acesso em: 16 nov. 2025. Disponível em: <https://blog.apify.com/python-playwright/>. Citado na página 40.

E-Commerce Update. *Artificial intelligence boosts e-commerce results in Brazil in 2024 and heats up the sector in 2025*. 2025. E-Commerce Update. Acesso em: 23 maio 2025. Disponível em: <https://www.ecommerceupdate.org/en/noticias/inteligencia-artificial-impulsiona-resultados-de->. Citado na página 17.

ELMAGARMID, A. K.; IPEIROTIS, P. G.; VERYKIOS, V. S. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, Institute of Electrical and Electronics Engineers (IEEE), v. 19, n. 1, p. 1–16, jan. 2007. ISSN 1041-4347. Disponível em: <http://dx.doi.org/10.1109/TKDE.2007.250581>. Citado 2 vezes nas páginas 18 e 26.

Equifax. *From Pain to Gain: Transforming Disparate Data Into a Revenue-Generating Resource*. 2023. Acesso em: 23 maio 2025. Disponível em: <https://www.equifax.com/business/blog/-/insight/article/from-pain-to-gain-transforming-disparate-data-into-a-revenue-generat>. Citado na página 18.

ERMAKOVICH, S. *Is Web Scraping Legal? A Guide for 2024*. 2025. Disponível em: https://hasdata.com/blog/is-web-scraping-legal. Acesso em: 10 out. 2025. Citado na página 22.

FERNANDES, D. *Marketplaces e sites de comparação de preços marcam novo comportamento do consumidor*. 2023. E-commerce Brasil. Acesso em: 23 maio 2025. Disponível em: <https://www.ecommercebrasil.com.br/noticias/marketplaces-comparacao-precos-comportamento-co> Citado na página 17.

GUAN, S. *BeautifulSoup vs Selenium: The Ultimate Showdown*. 2024. Disponível em: https://thunderbit.com/blog/beautifulsoup-vs-selenium. Acesso em: 10 out. 2024. Citado na página 21.

GUESDON, M. *Defining Web Scraping to Improve Mitigation*. 2024. Disponível em: https://antiscrapingalliance.org/defining-web-scraping-to-improve-mitigation-research-paper/. Acesso em: 10 out. 2024. Citado na página 21.

GUNTUPALLI, B. Asynchronous programming in java/python: A developer's guide. *International Journal of Emerging Research in Engineering and Technology*, ScienceTech Xplore, v. 3, n. 2, p. 70–78, dez. 2022. ISSN 3050-922X. Disponível em: <http://dx.doi.org/10.63282/3050-922X.IJERET-V3I2P108>. Citado na página 48.

GUSFIELD, D. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997. ISBN 9780511574931. Disponível em: <http://dx.doi.org/10.1017/cbo9780511574931>. Citado na página 28.

HAND, D.; CHRISTEN, P. A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing*, Springer Science and Business Media LLC, v. 28, n. 3, p. 539–547, abr. 2017. ISSN 1573-1375. Disponível em: <http://dx.doi.org/10.1007/s11222-017-9746-6>. Citado na página 32.

HARRIS, Z. S. Distributional structure. *Word*, Informa UK Limited, v. 10, n. 2–3, p. 146–162, ago. 1954. ISSN 2373-5112. Disponível em: <http://dx.doi.org/10.1080/00437956.1954.11659520>. Citado na página 25.

Hugging Face. *sentence-transformers/all-MiniLM-L6-v2*. 2024. Acesso em: 18 nov. 2025. Disponível em: <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>. Citado na página 37.

Iubenda. *Is Web Scraping Legal? What You Need to Know*. 2024. Disponível em: https://www.iubenda.c
is-web-scraping-legal-what-you-need-to-know. Acesso em: 10 out. 2024. Citado na página 22.

IVO, D. *E-commerce no Brasil: conheça os principais dados, o market share, o crescimento e as principais estatísticas, com atualização mensal!* 2025. [Online; accessed 2025-11-12]. Disponível em: <https://www.conversion.com.br/blog/relatorio-ecommerce-mensal/>. Citado na página 15.

JACCARD, P. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, v. 37, p. 547–579, 1901. Disponível em: <https://doi.org/10.5169/SEALS-266450>. Citado na página 57.

JACCARD, P. Nouvelles recherches sur la distribution florale. *Bulletin de la Société Vaudoise des Sciences Naturelles*, v. 44, p. 223–270, 1908. Disponível em: <https://doi.org/10.5169/seals-268384>. Citado na página 57.

JARO, M. A. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, v. 84, n. 406, p. 414–420, 1989. Disponível em: <https://doi.org/10.2307/2289924>. Citado 3 vezes nas páginas 28, 55 e 58.

JAYAN, J. *Ultimate Guide to Mastering Web Scraping for Data Extraction*. 2025. Acesso em: 23 maio 2025. Disponível em: <https://www.promptcloud.com/blog/guide-to-web-scraping/>. Citado na página 18.

JIANG, P.; CAI, X. A survey of text-matching techniques. *Information*, MDPI AG, v. 15, n. 6, p. 332, 6 2024. ISSN 2078-2489. Disponível em: <http://dx.doi.org/10.3390/info15060332>. Citado na página 33.

JOEVU. *How to build a web scraper with Python and Playwright*. 2023. Acesso em: 8 nov. 2025. Disponível em: <https://glinteco.com/en/post/how-to-build-a-web-scraper-with-python-and-pla>. Citado na página 48.

JONES, A. *Web scraping: Scraping AJAX and JavaScript websites*. 2025. Acesso em: 8 nov. 2025. Disponível em: <https://www.octoparse.com/blog/web-scraping-scraping-ajax-and-javascript-w
Citado na página 47.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd. ed. Stanford University, 2025. Online manuscript released August 24, 2025. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/>. Citado 2 vezes nas páginas 18 e 24.

KATHURIA, A.; GUPTA, A.; SINGLA, R. K. A review of tools and techniques for preprocessing of textual data. In: SINGH, V. et al. (Ed.). *Computational Methods and Data Engineering*. Singapore: Springer Singapore, 2021. p. 407–422. ISBN 978-981-15-6876-3. Disponível em: <https://doi.org/10.1007/978-981-15-6876-3_31>. Citado na página 41.

KAUFMAN, A. R.; KLEVS, A. Adaptive fuzzy string matching: How to merge datasets with only one (messy) identifying field. *Political Analysis*, Cambridge University Press (CUP), v. 30, n. 4, p. 590–596, out. 2021. ISSN 1476-4989. Disponível em: <http://dx.doi.org/10.1017/pan.2021.38>. Citado na página 28.

KAUR, A.; PRASHAR, D. Web scraping for product recommendations: A review of techniques and applications. *Journal of Computer Science*, Science Publications, v. 21, n. 6, p. 1425–1439, jun. 2025. ISSN 1549-3636. Disponível em: <http://dx.doi.org/10.3844/jcssp.2025.1425.1439>. Citado na página 18.

KIM, H. K.; KIM, H.; CHO, S. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. *Neurocomputing*, Elsevier BV, v. 266, p. 336–352, nov. 2017. ISSN 0925-2312. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2017.05.046>. Citado na página 29.

KÖPCKE, H. et al. Tailoring entity resolution for matching product offers. In: *Proceedings of the 15th International Conference on Extending Database Technology*. New York, NY, USA: ACM, 2012. p. 545–550. Citado na página 24.

KOUDAS, N.; MARATHE, A.; SRIVASTAVA, D. Flexible string matching against large databases in practice. In: _____. *Proceedings 2004 VLDB Conference*. Elsevier, 2004. p. 1078–1086. ISBN 9780120884698. Disponível em: <http://dx.doi.org/10.1016/B978-012088469-8.50094-2>. Citado na página 50.

KRANTZ, T.; JONKER, A. *O que É similaridade de Cosseno?* 2025. Disponível em: <https://www.ibm.com/br-pt/think/topics/cosine-similarity>. Citado na página 59.

KULYK, O. *Optimizing Web Scraping Speed in Python - Techniques and Best Practices*. 2024. Acesso em: 8 nov. 2025. Disponível em: <https://scrapingant.com/blog/fast-web-scraping-python>. Citado na página 48.

KYSELA, J. A comparison of text string similarity algorithms for poi name harmonisation. In: _____. *Articulated Motion and Deformable Objects*. Springer International Publishing, 2018. p. 121–130. ISBN 9783319945446. Disponível em: <http://dx.doi.org/10.1007/978-3-319-94544-6_12>. Citado na página 28.

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, v. 10, n. 8, p. 707–710, 1966. Disponível em: <https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf>. Citado na página 28.

LI, Y. et al. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, Association for Computing Machinery (ACM), v. 14, n. 1, p. 50–60, set. 2020. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/3421424.3421431>. Citado na página 34.

LUCIO, A. *E-commerce brasileiro movimentou R$ 100,5 bilhões no 1º semestre de 2025 - E-Commerce Brasil*. 2025. [Online; accessed 2025-11-12]. Disponível em: <https://www.ecommercebrasil.com.br/noticias/e-commerce-brasileiro-movimentou-r-1005-bilhoes-1o-semestre-de-2
Citado na página 15.

MACIENTE, R. Z. *Web Scraping e extração de dados em estruturas HTML com visualização em Interface Web: um estudo de caso*. 64 p. Monografia (Bacharelado em Ciência da Computação) — Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais –

Campus Formiga, Formiga, MG, 2025. Orientador: Prof. Me. Walace de Almeida Rodrigues. Citado na página 33.

MALAGA, K. B. K. An enhancement of the jaro-winkler fuzzy searching algorithm applied in library search engine. *Journal of Information Systems Engineering and Management*, Science Research Society, v. 10, n. 28s, p. 649–660, mar. 2025. ISSN 2468-4376. Disponível em: <http://dx.doi.org/10.52783/jisem.v10i28s.4369>. Citado na página 28.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. ISBN 978-0-521-86571-5. Disponível em: <https://nlp.stanford.edu/IR-book/>. Citado 2 vezes nas páginas 31 e 32.

MARCHANT, N. G.; RUBINSTEIN, B. I. P.; STEORTS, R. C. Bayesian graphical entity resolution using exchangeable random partition priors. *Journal of Survey Statistics and Methodology*, Oxford University Press (OUP), v. 11, n. 3, p. 569–596, jan. 2023. ISSN 2325-0992. Disponível em: <http://dx.doi.org/10.1093/jssam/smac030>. Citado na página 16.

MARTINS, P. et al. *Transformer-based Ranking Approaches for Keyword Queries over Relational Databases*. [S.l.], 2025. Disponível em: <https://arxiv.org/abs/2503.18768>. Citado na página 30.

MCKINNEY, W. pandas: a foundational python library for data analysis and statistics. In: *Proceedings of the 9th Python in Science Conference (SciPy 2011)*. [s.n.], 2011. Disponível em: <https://api.semanticscholar.org/CorpusID:61539023>. Citado 3 vezes nas páginas 36, 41 e 46.

Microsoft. *Playwright Python Documentation*. 2025. Acesso em: 8 nov. 2025. Disponível em: <https://playwright.dev/python/docs/>. Citado na página 47.

MIM, K. *Do's and Don'ts of Web Scraping*. 2021. Medium. Disponível em: https://medium.com/@dataj
and-donts-of-web-scraping-e4f9b2a49431. Acesso em: 10 out. 2024. Citado 3 vezes nas páginas 20, 21 e 22.

MUKHERJEE, S. *Levenshtein Distance: A Comprehensive Guide*. 2025. Disponível em: https://www.digitalocean.com/community/tutorials/levenshtein-distance-python. Acesso em: 10 out. 2024. Citado na página 27.

MüLLER, A.; KUWERTZ, A. A two-tire approach for organization name entity resolution. In: *Proceedings of the 11th International Conference on Data Science, Technology and Applications*. SCITEPRESS - Science and Technology Publications, 2022. p. 484–491. Disponível em: <http://dx.doi.org/10.5220/0011307000003269>. Citado na página 16.

NOVKOVIC, M. *Why is Knowing Your Competitors' Product Assortment Crucial in eCommerce?* 2025. Acesso em: 23 maio 2025. Disponível em: <https://www.price2spy.com/blog/competitor-product-assortment-analysis/>. Citado na página 17.

OK, J. *Top web scraping tools for scrapers*. 2025. Acesso em: 8 nov. 2025. Disponível em: <https://multilogin.com/blog/top-web-scraping-tools-for-scrapers/>. Citado na página 47.

OLIPHANT, T. E. *Guide to NumPy*. 2nd. ed. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2015. ISBN 151730007X. Disponível em: <ISBN978-1-5173-0007-4>. Citado na página 46.

PADRóN, M. *AI-powered web scraping with RAG*. 2025. Acesso em: 8 nov. 2025. Disponível em: <https://ardor.cloud/blog/ai-powered-web-scraping-with-rag>. Citado na página 48.

PAPADAKIS, G. et al. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys*, Association for Computing Machinery (ACM), v. 53, n. 2, p. 1–42, mar. 2020. ISSN 1557-7341. Disponível em: <http://dx.doi.org/10.1145/3377455>. Citado 2 vezes nas páginas 27 e 34.

PAPADAKIS, G. et al. Comparative analysis of approximate blocking techniques for entity resolution. *Proceedings of the VLDB Endowment*, Association for Computing Machinery (ACM), v. 9, n. 9, p. 684–695, maio 2016. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/2947618.2947624>. Citado 2 vezes nas páginas 18 e 26.

PAPADAKIS, G. et al. Jedai: The force behind entity resolution. In: _____. *The Semantic Web: ESWC 2017 Satellite Events*. Springer International Publishing, 2017. p. 161–166. ISBN 9783319704074. Disponível em: <http://dx.doi.org/10.1007/978-3-319-70407-4_30>. Citado na página 34.

PaymentsCMI. *Visão geral do mercado de comércio eletrônico no Brasil*. 2025. [Online; accessed 2025-11-12]. Disponível em: <https://paymentscmi.com/insights/mercado-comercio-eletronico-r>. Citado na página 15.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, JMLR.org, v. 12, n. null, p. 2825–2830, nov. 2011. ISSN 1532-4435. Disponível em: <https://dl.acm.org/doi/pdf/10.5555/1953048.2078195>. Citado na página 37.

PEETERS, R.; BIZER, C. Using chatgpt for entity matching. In: _____. *New Trends in Database and Information Systems*. Springer Nature Switzerland, 2023. p. 221–230. ISBN 9783031429415. Disponível em: <http://dx.doi.org/10.1007/978-3-031-42941-5_20>. Citado na página 34.

PEETERS, R.; STEINER, A.; BIZER, C. *Entity Matching using Large Language Models*. OpenProceedings.org, 2025. Disponível em: <https://doi.org/10.48786/edbt.2025.42>. Citado na página 16.

PHOENIX, J. *The Advantages Disadvantages of Web Scraping Data*. 2024. [Online; accessed 2025-11-12]. Disponível em: <https://understandingdata.com/posts/the-advantages-disadvantages-
>. Citado 2 vezes nas páginas 15 e 17.

Pricer24. *Pricing Intelligence for Retailers: A Comprehensive Guide*. 2023. Acesso em: 23 maio 2025. Disponível em: <https://pricer24.com/blog/pricing-intelligence-for-retailers/>. Citado na página 17.

PRIMPELI, A.; BIZER, C. Impact of the characteristics of multi-source entity matching tasks on the performance of active learning methods. In: _____. *The Semantic Web*. Springer International Publishing, 2022. p. 113–129. ISBN 9783031069819. Disponível em: <http://dx.doi.org/10.1007/978-3-031-06981-9_7>. Citado na página 18.

PromptCloud. *Compliance and Data Governance*. 2025. Disponível em: https://www.promptcloud.com/data-governance/. Acesso em: 25 out. 2025. Citado 2 vezes nas páginas 22 e 23.

Python Software Foundation. *Python Language Reference*. [S.l.], 2024. Acesso em: 8 nov. 2025. Disponível em: <https://docs.python.org/3/reference/index.html>. Citado na página 46.

RAGHAVAN, V. V.; WONG, S. K. M. A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, Wiley, v. 37, n. 5, p. 279–287, set. 1986. ISSN 1097-4571. Disponível em: <http://dx.doi.org/10.1002/(SICI)1097-4571(198609)37:5<279::AID-ASI1>3.0.CO;2-Q>. Citado na página 18.

RAGKHITWETSAGUL, C.; KRINKE, J.; CLARK, D. A comparison of code similarity analysers. *Empirical Software Engineering*, Springer Science and Business Media LLC, v. 23, n. 4, p. 2464–2519, out. 2017. ISSN 1573-7616. Disponível em: <http://dx.doi.org/10.1007/s10664-017-9564-7>. Citado na página 27.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019. p. 3982–3992. Disponível em: <https://doi.org/10.18653/v1/D19-1410>. Citado 5 vezes nas páginas 18, 30, 31, 35 e 60.

RIVAS-SáNCHEZ, M. et al. Product matching to determine the energy efficiency of used cars available at internet marketplaces. In: _____. *Soft Computing for Sustainability Science*.

Springer International Publishing, 2017. p. 203–215. ISBN 9783319623597. Disponível em: <http://dx.doi.org/10.1007/978-3-319-62359-7_10>. Citado na página 16.

Salishsea Consulting. *The Complete Guide to Ethical Web Scraping for Conscious Businesses*. 2024. Disponível em: https://www.salishseaconsulting.com/blog/the-complete-guide-to-ethical-web-scraping-for-conscious-businesses/. Acesso em: 10 out. 2024. Citado na página 22.

SALTON, G.; WONG, A.; YANG, C. S. A vector space model for automatic indexing. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 18, n. 11, p. 613–620, nov. 1975. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/361219.361220>. Citado na página 59.

SHAMNDY, Z. *Playwright vs Selenium*. 2025. Scrapfly. Acesso em: 8 nov. 2025. Disponível em: <https://scrapfly.io/blog/posts/playwright-vs-selenium>. Citado 2 vezes nas páginas 47 e 48.

SHARMA, N. *Importance of Distance Metrics in Machine Learning Modelling*. 2019. Medium. Disponível em: https://medium.com/data-science/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d. Acesso em: 10 out. 2024. Citado 2 vezes nas páginas 29 e 30.

SHENOY, S. *Elegant text pre-processing with NLTK in Sklearn pipeline*. 2022. Disponível em: https://towardsdatascience.com/elegant-text-pre-processing-with-nltk-in-sklearn-pipeline-d6fe18b91eb8. Acesso em: 10 out. 2024. Citado 2 vezes nas páginas 23 e 25.

SIINO, M.; TINNIRELLO, I.; CASCIA, M. L. Is text preprocessing still worth the time? a comparative survey on the influence of popular preprocessing methods on transformers and traditional classifiers. *Information Systems*, Elsevier BV, v. 121, p. 102342, mar. 2024. ISSN 0306-4379. Disponível em: <http://dx.doi.org/10.1016/j.is.2023.102342>. Citado na página 50.

SILGE, J.; ROBINSON, D. *Text Mining with R: A Tidy Approach*. 1st. ed. O'Reilly Media, Inc., 2017. ISBN 1491981652. Disponível em: <https://www.tidytextmining.com/>. Citado 2 vezes nas páginas 25 e 29.

SINGLA, P.; DOMINGOS, P. Entity resolution with markov logic. In: *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006. p. 572–582. ISSN 1550-4786. Disponível em: <http://dx.doi.org/10.1109/ICDM.2006.65>. Citado na página 26.

SOUZA, J. P. de. *Protótipo de uma Plataforma de busca de Preços na Web*. 83 p. Monografia (Bacharelado em Ciência da Computação) — Instituto Federal de Educação, Ciência e

Tecnologia de Minas Gerais – Campus Formiga, Formiga, MG, 2025. Orientador: Prof. Dr. Bruno Ferreira. Citado 2 vezes nas páginas 33 e 34.

SOVIERSOVSKI, N. *Webshoppers: Um panorama atual do e-commerce brasileiro*. 2025. [Online; accessed 2025-11-12]. Disponível em: <https://ghfly.com/artigo-webshoppers-panorama-do-ec
>. Citado 2 vezes nas páginas 15 e 17.

SPERBER, A. *Boost Your Online Electronics Store with Top Google Shopping Strategies*. 2023. Acesso em: 15 nov. 2025. Disponível em: <https://unitedads.com/blog/the-best-google-shopping-strategies-for-electronics-online-stores/>. Citado na página 33.

STEFANIDIS, K. et al. Entity resolution in the web of data. In: *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 2014. (WWW '14), p. 203–204. Disponível em: <http://dx.doi.org/10.1145/2567948.2577263>. Citado 2 vezes nas páginas 26 e 27.

STRINGER, J. *How to Optimise a Product Data Feed for Google Merchant Center*. 2025. Acesso em: 15 nov. 2025. Disponível em: <https://stringerseo.co.uk/content/how-to-optimise-a-produc
>. Citado na página 33.

STSIOPKINA, M. *Selenium vs BeautifulSoup: Which is Better for Web Scraping?* 2024. Disponível em: https://oxylabs.io/blog/selenium-vs-beautifulsoup. Acesso em: 10 out. 2024. Citado na página 21.

SUBRAMANIY, V.; PANDIAN, S. C. A complete survey of duplicate record detection using data mining techniques. *Information Technology Journal*, Science Alert, v. 11, n. 8, p. 941–945, jul. 2012. ISSN 1812-5638. Disponível em: <http://dx.doi.org/10.3923/itj.2012.941.945>. Citado na página 26.

TGNData. *Why Competitor Assortment Data Matters*. 2024. Acesso em: 23 maio 2025. Disponível em: <https://tgndata.com/why-competitor-assortment-matters/>. Citado na página 17.

TORCATO, S. *Top 9 Web Scraping Challenges in E-Commerce Data*. 2023. Acesso em: 23 maio 2025. Disponível em: <https://www.datahen.com/blog/web-scraping-challenges-in-ecommerce/
>. Citado na página 18.

VOGEL, T. et al. Reach for gold: An annealing standard to evaluate duplicate detection results. *Journal of Data and Information Quality*, Association for Computing Machinery (ACM), v. 5, n. 1–2, p. 1–25, set. 2014. ISSN 1936-1963. Disponível em: <http://dx.doi.org/10.1145/2629687>. Citado na página 64.

WALDFOGEL, J. *Amazon Self-preferencing in the Shadow of the Digital Markets Act*. [S.l.], 2024. Disponível em: <http://dx.doi.org/10.3386/w32299>. Citado na página 33.

WANG, W. et al. Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2020. (NIPS '20). ISBN 9781713829546. Disponível em: <https://doi.org/10.5555/3495724.3496209>. Citado 2 vezes nas páginas 56 e 60.

WINKLER, W. E. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: AMERICAN STATISTICAL ASSOCIATION. *Proceedings of the Survey Research Methods Section*. Alexandria, VA, 1990. p. 354–359. Disponível em: <https://api.semanticscholar.org/CorpusID:54580585>. Citado 2 vezes nas páginas 55 e 58.

YULIANTON, H.; SANTI, R. C. N. Product matching using sentence-bert: A deep learning approach to e-commerce product deduplication. *Engineering and Technology Journal*, Everant Journals, v. 09, n. 12, p. 5659–5664, dez. 2024. ISSN 2456-3358. Disponível em: <http://dx.doi.org/10.47191/etj/v9i12.14>. Citado na página 16.

APÊNDICE A – Exemplos de *Clusters*

A.1 SBERT

Foram coletados alguns *clusters* de amostra a partir da técnica SBERT, sendo possível observar o agrupamento de produtos distintos, mas que possuem uma semelhança semântica:


```
{
    "grupo_id": 619,
    "produtos": [
        {
            "id": 6510,
            "nome": "chocolate ao leite com biscoito duo cookie
               classic pacote",
            "plataforma": "abc"
        },
        {
            "id": 4458,
            "nome": "chocolate ao leite com biscoito duo cookie
               classic pacote",
            "plataforma": "rena"
        },
        {
            "id": 3000,
            "nome": "biscoito cookie recheio chocolate ao leite
               e gotas de chocolate choco cookie pacote",
            "plataforma": "abc"
        },
        {
            "id": 2560,
            "nome": "biscoito cookie sabor chocolate com gotas
               de chocolate ao leite classic pacote",
            "plataforma": "abc"
        }
    ]
},

{
    "grupo_id": 646,
```

```
    "produtos": [
        {
            "id": 4403,
            "nome": "doce leite sao lourenco diet coco",
            "plataforma": "rena"
        },
        {
            "id": 6354,
            "nome": "doce de leite c coco diet sao lourenco",
            "plataforma": "abc"
        },
        {
            "id": 4407,
            "nome": "doce de leite diet sao lourenco",
            "plataforma": "rena"
        },
        {
            "id": 5613,
            "nome": "doce de leite com ameixa diet sao lourenco",
            "plataforma": "abc"
        },
        {
            "id": 5643,
            "nome": "doce de leite c coco diet sao lourenco",
            "plataforma": "abc"
        },
        {
            "id": 4423,
            "nome": "doce leite sao lourenco diet coco",
            "plataforma": "rena"
        },
        {
            "id": 4735,
            "nome": "leite condensado diet doces sao lourenco
              lata",
            "plataforma": "rena"
        },
        {
            "id": 6190,
            "nome": "leite condensado diet doces sao lourenco
              lata",
            "plataforma": "abc"
```

```
        },
        {
            "id": 6189,
            "nome": "doce de leite com ameixa diet sao lourenco",
            "plataforma": "abc"
        }
    ]
},
{
    "grupo_id": 826,
    "produtos": [
        {
            "id": 3472,
            "nome": "oleo composto de soja e oliva tradicional",
            "plataforma": "carrefour"
        },
        {
            "id": 5211,
            "nome": "oleo composto tradicional",
            "plataforma": "super so"
        },
        {
            "id": 6191,
            "nome": "oleo composto soja e oliva tradicional
               lata",
            "plataforma": "abc"
        },
        {
            "id": 6416,
            "nome": "oleo composto de soja e oliva tradicional
               lata",
            "plataforma": "abc"
        }
    ]
}
```


A.2 *Jaro-Winkler*

Foram coletados alguns *clusters* de amostra a partir da técnica *Jaro-Winkler*, sendo possível observar o agrupamento de produtos que compartilham do mesmo prefixo, mas que na prática são produtos bem distintos.

```
{
    "grupo_id": 227,
    "produtos": [
        {
            "id": 5868,
            "nome": "cafe coffee++ capsula",
            "plataforma": "abc"
        },
        {
            "id": 6467,
            "nome": "cafe coffee++ em graos",
            "plataforma": "abc"
        },
        {
            "id": 6468,
            "nome": "cafe coffee++ em graos",
            "plataforma": "abc"
        },
        {
            "id": 5869,
            "nome": "cafe coffee++ drip classico",
            "plataforma": "abc"
        },
        {
            "id": 6480,
            "nome": "cafe coffee++ moido",
            "plataforma": "abc"
        },
        {
            "id": 5863,
            "nome": "cafe coffee++ capsula",
            "plataforma": "abc"
        },
        {
            "id": 6478,
            "nome": "cafe coffee++ capsula chapada de minas",
            "plataforma": "abc"
        },
        {
            "id": 6482,
            "nome": "cafe coffee++ drip classico",
```

```
            "plataforma": "abc"
        },
        {
            "id": 6466,
            "nome": "cafe coffee++ moido",
            "plataforma": "abc"
        },
        {
            "id": 5867,
            "nome": "cafe coffee++ capsula chapada de minas",
            "plataforma": "abc"
        },
        {
            "id": 6479,
            "nome": "cafe coffee++ capsula",
            "plataforma": "abc"
        },
        {
            "id": 6484,
            "nome": "cafe coffee++ cerrado mineiro moido",
            "plataforma": "abc"
        },
        {
            "id": 6483,
            "nome": "cafe coffee++ classico moido",
            "plataforma": "abc"
        },
        {
            "id": 6474,
            "nome": "cafe coffee++ capsula",
            "plataforma": "abc"
        }
    ]
},
{
    "grupo_id": 278,
    "produtos": [
        {
            "id": 6299,
            "nome": "creme de leite",
            "plataforma": "abc"
        },
```

```
        {
            "id": 4616,
            "nome": "creme leite fresco",
            "plataforma": "rena"
        },
        {
            "id": 252,
            "nome": "creme de leite fresco sem lactose",
            "plataforma": "carrefour"
        },
        {
            "id": 19,
            "nome": "creme de leite fresco sem lactose",
            "plataforma": "carrefour"
        },
        {
            "id": 1763,
            "nome": "creme de leite leve r",
            "plataforma": "rena"
        },
        {
            "id": 1661,
            "nome": "creme de leite leve caixa",
            "plataforma": "rena"
        },
        {
            "id": 3440,
            "nome": "creme de leite pasteurizado",
            "plataforma": "carrefour"
        },
        {
            "id": 6197,
            "nome": "creme de leite",
            "plataforma": "abc"
        },
        {
            "id": 106,
            "nome": "creme de leite",
            "plataforma": "carrefour"
        },
        {
            "id": 202,
```

```
            "nome": "creme de leite fresco",
            "plataforma": "carrefour"
        },
        {
            "id": 144,
            "nome": "creme de leite leve",
            "plataforma": "carrefour"
        },
        {
            "id": 2593,
            "nome": "creme de leite leve caixa",
            "plataforma": "abc"
        },
        {
            "id": 218,
            "nome": "creme de leite",
            "plataforma": "carrefour"
        }
    ]
},
```